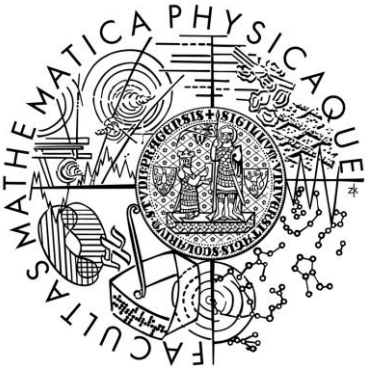
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Lukáš Šurín

RPGcraft

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Petr Hnětynka, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná Informatika

Praha rok 2013

Ďakujem pánovi RNDr. Petrovi Hnětynkovi, Ph.D. za podporu, trpezlivosť a pomoc preukázanú pri vedení tejto práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... podpis

Názov práce: RPGcraft

Autor: Lukáš Šurín

Katedra / Ústav: Katedra distribuovaných a spoľahlivých systémov

Vedúci bakalárskej práce: RNDr. Petr Hnětynka, Ph.D.

Abstrakt: Cieľom tejto práce je vytvorenie herného prostredia, ktoré by zdieľalo podobnosti s hrou Minecraft s pridanými RPG prvkami. Prostredie je v práci zanalyzované s popisom vlastností, ktoré by mal projekt obsahovať pre naplnenie stanovených cieľov. Obsahuje vypracovaný návrh a implementáciu projektu s využitím techník pre efektívny a korektný beh. Súčasť práce je popis možných spôsobov rozšírenia hier a ukázanie použitia knižnice LuaJ na vykonanie lua skriptov. Práca popisuje dve vytvorené hry k nej priložené a porovnáva schopnosti vytvoreného herného prostredia s ďalšími rôznymi titulmi, s ktorými je ju vhodné porovnávať.

Kľúčové slová: Hry, Minecraft, Hra na hrdinov, Lua

Title: RPGcraft

Author: Lukáš Šurín

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Petr Hnětynka, Ph.D.

Abstract: The aim of this work is to create game environment with added RPG elements which is sharing similarities with the Minecraft. We analyze the game environment further describing the properties that the project should include for the fulfillment of objectives. It contains design and implementation of the game using techniques for efficient and correct running. Project provides a description of possible extensions and shows the usage of LuaJ library for the execution of Lua scripts. The resulting state of the project describes two created games and compares the ability of the established gaming environment with a variety of others which it is appropriate to compare.

Keywords: Games, Minecraft, RPG, Lua

Obsah

1 Úvod 4

2 Analýza hry 7

2.1 Zobrazenie sveta 7

2.2 Simulovanie sveta 8

2.3 Spracovanie sveta 8

2.4 Spracovanie vstupu 9

2.5 Spracovanie objektov 10

2.6 Skriptovanie 10

2.7 Možnosť modifikácie 11

3 Herný návrh 13

3.1 Samotné hranie 13

3.1.1 Hra v reálnom čase 13

3.1.2 Ovládanie 14

3.2 Možnosti sveta 14

3.2.1 Nekonečné svety 14

3.2.2 Multi poschodia 15

3.2.3 Procedurálne generovanie 15

3.3 Možnosti hráča 15

3.3.1 Interakcie s cudzími entitami 16

3.3.2 Prístup k vlastnostiam, úlohám, batohu 17

3.3.3 Vytváranie predmetov 17

3.3.4 Interakcie so svetom 17

3.4 Príbeh 17

3.4.1 Možnosti tvorenia úloh 17

3.5 Rozšírenie hry 18

3.5.1 Skriptovanie 18

4 Implementácia a použité technológie 20
4.1 Programovací jazyk 20
4.2 Spracovávanie hry v reálnom čase 20
4.2.1 Animačný cyklus 20
4.2.2 Aktívne renderovanie 22
4.3 Spravovanie hracieho plánu 23
4.3.1 Aktualizácia plánu/mapy 23
4.3.2 Vykreslenie plánu/mapy 24
4.4 Spravovanie objektov 24
4.4.1 Objekty v zdrojových/resource súboroch 25
4.4.2 Objekty spravované hracím plánom 25
4.5 Spracovanie užívateľského vstupu 26
4.5.1 Globálny ovládač pre vstup z klávesnice 26
4.5.2 Spracovanie udalostí z myši 26
4.6 Spracovávanie a vykonávanie udalostí 27
4.6.1 Udalosti v príbehu 27
4.6.2 Udalosti v konverzáciách 27
4.6.3 Udalosti v efektoch 28
4.7 Užívateľské prostredie 28
4.7.1 Menu s pomocou SWING 28
4.7.2 Menu priamo v hre 28
4.8 Možnosti rozšírenia hry 29
4.8.1 Zdrojové XML súbory 29
4.8.2 Plugin systém 31
5 Ukážkové projekty 36
5.1 Ukážková hra 36
5.2 Ukážková testovacia hra 39

6 Príbuzné práce 41
6.1 Minecraft, 2009 41
6.2 Dwarf Fortress, 2006 42
6.3 ElderScrolls séria (Morrowind) 42
7 Záver 44
7.1 Budúce práce 44
8 Zoznam použitej literatúry 46
9 Zoznam obrázkov 48
Obsah priloženého CD 49

1 Úvod

Hry sa v posledných rokoch stali neoddeliteľnou súčasťou nášho života, čím sa podporila tvorba rôznych nových zaujímavých titulov. V tejto práci sa zaoberáme návrhom a tvorbou hry typu Sandbox obohatenej o prvky hier typu RPG[1]. Hra je inšpirovaná hrou Minecraft [2], pričom je ju možné rozširovať pomocou pluginov a skriptov.

1.1 Čo myslíme žánrom RPG?

RPG (Role Playing Game) je typ hry spojený s možnosťou ovládania svojho vlastného hráča (niekedy viacero hráčov). S postavou sa pohybujeme po danej mape, objavujeme nové miesta a získavame predmety, ktorými môžeme zlepšovať vlastnosti ovplyvňujúce rôzne akcie v hre (spravujeme svoju postavu). Taktiež interagujeme s okolím, v ktorom ničíme nepriateľov a plníme úlohy, za ktoré môžeme získať odmenu.

Historicky si žáner od vzniku v roku 1970 prešiel rôznymi obmenami (graficky aj funkčne), a tak ako bolo očakávané, vznikali rôzne kombinácie s inými druhmi žánrov (akčne RPG, taktické RPG,...). V dnešnej dobe existuje mnoho titulov s RPG prvkami, čím sa RPG stalo jedným z najobľúbenejších [1] žánrov v histórii hier.

1.2 Základný princíp Sandbox/Minecraft.

Sandbox bol relatívne zabudnutý žáner až do uvedenia hry Minecraft. Tento žáner zahrňuje otvorený svet s možnosťou interakcie prostredníctvom tvorby. Hlavný rozdiel s RPG žánrom je ten, že Sandbox interaguje viac so svetom samotným a nerieši správu postavy pre stanovené ciele hry.

V roku 2009 vyšiel výtvor švédskeho programátora Marcussa Perssona s názvom Minecraft [2]. Je to nezávislá[2] 3D hra typu sandbox, vytvorená v prostredí Java, ktorá sa skladá výhradne iba z kociek. Postava má možnosť prechádzať rozsiahlym, procedurálne generovaným svetom, ktorý môžeme pretvárať podľa

[1] Hra na hrdiny

[2] Nezávislé alebo tiež Indie je označenie pre hry, ktoré sú vytvorené skupinou ľudí, alebo jednotlivcom bez vydavateľa.

našich potrieb, v čom nám pomáhajú rôzne nástroje, ktoré sa dajú v hre vytvoriť. Zameranie sa pohybuje od dobrodružného objavovania sveta, kreatívneho vytvárania pixelového umenia [3] až po vytváranie rôznych prístrojov. Možnosti sú ešte doplnené pridaním obvodov vydávajúce impulzy, ktoré samovoľne aktualizujú niektoré predmety v hre. Obvody sa umiestňujú do priestoru. Pre vypísané vlastnosti má hra iba niekoľko megabajtov a je v neustálom vývoji, čím užívateľ získava prístup k novým aktualizáciám s novým obsahom. Ako nevýhodu je možné poukázať na absenciu hlbších RPG prvkov a to, že hráč nemá sám prístup k zmene, či pridaniu nového obsahu pomocou rozšírenia[3]. Minecraft je v tejto dobe globálne uznávaná hra, ktorá má široký úspech [4].

1.3 Cieľ práce

Naša práca má za úlohu vytvoriť „prázdne" prostredie pre hru podobnú Minecraft-u s akčnými prvkami RPG, ktoré chýbali v originálnej hre (úlohy, vlastnosti hráča a nimi ovplyvnené akcie, konverzácie, ...) a ktorá navyše bude a) rozšíriteľná skoro vo všetkých smeroch prostredníctvom XML súborov; b) rozšíriteľná pomocou pluginov; c) schopná reagovať na udalosti vyvolané skriptami. Popri prázdnemu prostrediu vytvárame aj ukážkovú hru využívajúcu XML súbory a pluginy.

Na rozdiel od Minecraft-u bude hra v 2D vytvorená iba na základnej platforme JAVA bez akýchkoľvek vykresľovacích knižníc. V hre bude chýbať tvorba obvodov, keďže už dosť komplikovaný systém stavby takýchto obvodov by bol v 2D svete ešte menej prehľadnejší. Hra neobsahuje automatické generovanie herného sveta, ktoré je ponechané ako jedna z možných budúcich smerov vývoja hry.

1.4 Štruktúra práce

V prvom oddieli sa budeme venovať analýze hry. Stanovíme ciele hry a preberieme, akými spôsobmi sa dajú riešiť jednotlivé stanovené ciele a problémy, ktoré môžu pri nich vzniknúť.

[3] Existuje vytvorený balíček na rozšírenie hry, ktorý však nepochádza od tvorcov Minecraft-u, ale vytvorila ho iná skupina ľudí, a tak môžu od tohto balíčku rýchlo upustiť.

Druhý oddiel sa venuje popisu herného návrhu, ktorý chceme implementovať. Rozoberieme súčasti takéhoto návrhu s vybranými spôsobmi na realizáciu. Máme v ňom rozpísanú štruktúru projektu a možnosti rozšírenia.

Tretí oddiel popisuje architektúru našej implementácie, v ktorej je rozpísané, akým spôsobom sú jednotlivé celky poprepájané. Nachádza sa tu popis štruktúry týchto celkov.

V štvrtej kapitole popisujeme ukážkové projekty, ktoré demonštrujú možnú implementáciu hier.

V piatom oddieli popisujeme praktickosť práce a to tak, že porovnávame našu prácu s dostupnými príbuznými prácami, ktoré sú jej čo najbližšie. Vypíšeme, čo jednotlivé práce oproti našej práci obsahujú resp. neobsahujú.

V záverečnej kapitole hodnotíme prácu a popisujeme či hra splňuje stanovené ciele. Tiež tu popisujeme možnosti, ako by hru bolo možné zlepšiť v budúcnosti.

2 Analýza hry

V tejto kapitole sa zameriavame na zanalyzovanie hry ako celku. Vymenujeme základné schopnosti, ktoré by mala hra obsahovať a vykonávať nato, aby spĺňala cieľ práce. U niektorých vypíšeme spôsoby, akými sa dá hra zrealizovať, s popisom prečo sme sa rozhodli pre takýto krok a s porovnaním pre ďalšie rôzne prístupy, ktoré by sa dali použiť. Keďže vytvárame sandbox hru s prvkami RPG, je dôležité uviesť, že bude obsahovať základné možnosti typické pre obidva žánre.

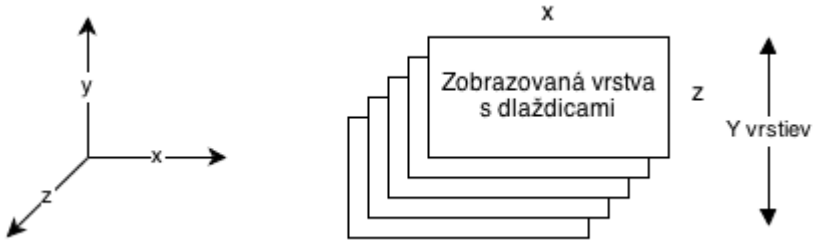
2.1 Zobrazenie sveta

Pre návrh hracieho jadra je dôležitou súčasťou spôsob, akým budeme vykresľovať svet na hraciu obrazovku. Hry môžu byť vykresľované dvoma spôsobmi, ktoré zahŕňajú *dvojrozmernú* a *trojrozmernú* mapu.

Originálny titul Minecraft je trojrozmerná hra zobrazovaná z prvej osoby. RPG hry bývajú trojrozmerné s možnosťou otáčania kamery. Výhodou je krajšie, zaujímavejšie a realistickejšie zobrazenie, ktoré skoro úplne vystriedalo dvojrozmerný prístup. Nevýhodou je vyššia obtiažnosť implementovania a nutnosť využitia rôznych 3D knižníc a modelov.

Dvojrozmerný prístup je ľahšou alternatívou, ktorá väčšinou zobrazuje svet z vtáčej perspektívy alebo izometricky. Objekty sú obyčajné obrázky, kde mapa je zložená z tzv. dlaždíc. V tejto dobe sa upúšťa od takýchto hier kvôli neatraktívnosti.

V našej hre si vyberáme kombináciu dvoch prístupov. Mapa bude trojrozmerná s možnosťou pohybu v troch rôznych smeroch. Zobrazenie takejto mapy bude dvojrozmerné, vykreslením jednej aktuálnej vrstvy. Takýto typ nazveme dvojrozmerné zobrazenie trojrozmerného sveta (obrázok 2.1). Výber sme učinili hlavne kvôli jednoduchšej implementácii hry, bez nutnosti hľadať 3D modely do nášho projektu. Ponechanie trojrozmerného sveta nám umožní budúce prepracovanie na tretí rozmer.



Obrázok 2.1 : Predstava 2D zobrazovania 3D mapy

2.2 Simulovanie sveta

Pred spracovaním sveta je nutné rozhodnúť, ako ho budeme simulovať. Hry bývajú rozdelené do viacerých žánrov, ktoré sú väčšinou priamo spojené aj s nejakou možnosťou simulácie, ktorá dáva pre žáner zmysel. Svet môžeme simulovať troma spôsobmi a to *reálne, ťahovo* a kombináciou týchto dvoch.

Reálne simulovanie dáva možnosť reagovať na hru simultánne s dianím v hre. Výhodou takéhoto prístupu je dynamickejšie poňatie hry, čo by mal každý akčnejší žáner obsahovať. Tento prístup taktiež poskytuje dodatočné výzvy a nutnosť organizovania akcií. Hlavnou nevýhodou je nutnosť aktualizácie všetkých objektov v hre, čo pri nesprávnom programovaní vedie k zníženiu rýchlosti hry.

V ťahovom štýle je hra rozdelená do ťahov, v ktorej môžeme zanalyzovať naše možnosti, a podľa nich vykonať akcie. Po ukončení ťahu hráča sú na rade iné objekty, ktoré vykonajú svoje pohyby. Výhodou ťahových hier je možnosť premyslieť, čo chceme vykonať a tým pádom umožniť hráčovi taktizovať. Aktualizácie objektov prebiehajú postupne po ťahoch, čím rýchlosť vykonania ťahu závisí od výkonu. Taktiež nie je až tak braná do úvahy rýchlosť hry. Hlavnou nevýhodou je práve samotný ťahový štýl, ktorý nedáva pocit akčnosti.

Pre potreby našej hry je treba uvážiť, že vytvárame Sandbox hru s akčnými RPG prvkami. Sandbox žáner umožňuje pohyb po širokom svete a ťahový prístup by nám bránil v tomto zážitku s nedostatočnou formou skutočnosti.

2.3 Spracovanie sveta

Medzi spracovanie sveta patrí to, ako ho budeme reprezentovať. Svet sa dá reprezentovať viacerými spôsobmi, kde základným spracovaním vo väčšine hier býva obmedzená veľkosť mapy, ktorá je rozdelená do tzv. l*evelov,* ktoré sú tvorené hernými dizajnérmi. Výhodou takéhoto prístupu je väčší dôraz na tvorbu mapy a detailov, čím hra príde hráčovi zaujímavejšia a reálnejšia. Nevýhodou je obmedzenosť čistého hracieho času, za ktorý hru prejdeme až do cieľa. Taktiež je nutné podotknúť, že na vytvorenie pútavého príbehu pre veľké projekty je potreba skupinu dizajnérov.

Ďalším spôsobom je tiež obmedzená mapa, ako je tomu v predchádzajúcom prípade s možnosťou opakovaného hrania a pozmenenou mapou či pozmenenými cieľmi a obtiažnosťou. Výhody má hra rovnaké, pričom dodatočne predlžujeme hrací

čas. Hlavnou nevýhodou je vytvoriť taký systém, ktorý podporuje tvorbu rôznej mapy pri opakovanom hraní, čo väčšinou obnáša tvorbu procedurálneho generátora máp.

Posledným zo spôsobov je vytvoriť nekonečnú mapu, ktorá sa vytvára postupne. Možnosti takéhoto prístupu umožňuje maximálne predĺženie hracieho času. Nevýhodou je nutnosť implementovania určitého generátora mapy, ktorý by generoval rôzne neopakujúce sa časti.

V našom projekte využijeme posledný spôsob doplnený o spracovanie nekonečného sveta ako do šírky, tak aj do hĺbky, keďže chceme napodobiť vlastnosti hry Minecraft v čo najviac smeroch. Taktiež chceme rozšíriť repozitár týchto druhov hier kvôli existencii mála titulov s nekonečným prístupom.

2.4 Spracovanie vstupu

Každý hrací žáner má určitý typ ovládania, ktorý preňho dáva zmysel. Existujú dva hlavné typy ovládania, a to pomocou klávesnice a myši.

Pri akčných hrách, strielačkách a hrách spoliehajúcich na rýchle reakcie treba také spracovanie vstupu, ktoré rýchlo a spoľahlivo reaguje a odpovedá na vstupy. V tomto prípade to je ovládanie skrze klávesnicu. Ovládanie pomocou klávesnice predstavuje rýchly spôsob hlavne kvôli existencii viacerých tlačidiel blízko seba, ktoré môžu vykonávať rôzne definované akcie. Takýto spôsob je tiež ľahšie prepracovateľný podľa aktuálneho trendu na konzolovú hru. Hlavnou nevýhodou je nezvyk hráča na používanie klávesnice ako primárneho ovládacieho nástroja.

Pri ležérnejších hrách ako sú ťahové stratégie, adventúry, a pod., či pri ovládacích prvkoch stačí nenáročné ovládanie so spoľahlivou reakciou na vstup. Tento spôsob najlepšie vystihuje použitie myši. Výhodou využívania myši je možnosť presnejších operácií. Nevýhodou sú pomalšie hracie reakcie, kvôli čomu sa v hráčskej komunite skôr preferuje klávesnica.

Pre naše potreby treba brať do úvahy, že hra bude obsahovať pomimo hlavnej hry aj grafické komponenty, pri ktorých je praktickejšie využívať myš. Kvôli tomu sa rozhodujeme pre kombináciu obidvoch s väčším dôrazom na klávesnicu.

2.5 Spracovanie objektov

Objekty a možnosti spracovania objektov by mali byť jednoduché natoľko, aby sa dalo s nimi čo najľahšie manipulovať. Existuje mnoho prístupov, ako hry riešia problémy s objektmi, ktoré zahŕňajú kedy, kde a ako tieto objekty aktualizovať či zobrazovať. Popísané sú kombinácie riešení týchto jednotlivých problémov.

Prvým riešením je združovať objekty v jednotlivých kúskoch mapy a pri načítaní a ukladaní plánu sa uložia iba tie, ktoré obsahuje. Objekty sa pri takomto riešení aktualizujú pri danom načítanom kúsku mapy. Zobrazujú sa iba tie vo viditeľnej vzdialenosti. Výhodou je vytvorená štruktúra pre objekty, pre ktoré dokážeme vždy určiť, na akom mieste sa nachádzajú. Nevýhodou je vyhľadanie konkrétneho objektu, kvôli čomu musíme prehľadať všetky kúsky mapy.

Jedným z riešení je udržovať objekty v určitej dátovej štruktúre, ktorá by podporovala rôzne usporiadania objektov, zistenie kolízií medzi objektmi, ale aj ľahké načítanie a ukladanie do nej. Pre tieto dôvody býva využívaná štruktúra *Octree*, ktorá našla svoje uplatnenie hlavne v 3D hrách, kvôli čomu je zbytočné takéto riešenie využívať.

Ďalším riešením je združovanie všetkých objektov v jednom liste, do ktorého môžeme ľahko pridávať, ale aj odstraňovať. Objekty v takomto prípade združujú informácie, medzi ktorými figuruje aj to, v ktorej mape sa objekt nachádza spolu s presnými pozíciami vo svete. Hlavnou výhodou je jednoduchosť takéhoto riešenia. Nevýhodou je veľké množstvo objektov v tomto liste a následné aktualizovanie a zisťovanie kolízií medzi objektmi, ktoré by doviedlo na kvadratickú zložitosť.

V našej hre využijeme posledné riešenie, hlavne kvôli nekomplikovateľnosti riešenia pre zvolenú 2D hru. Popísané problémy pri takomto prístupe budeme riešiť obmedzením objektov, nad ktorými vykonávame operácie, viz sekcia 4.3.1.

2.6 Skriptovanie

Možností ako ovplyvňovať dej v hre sa typicky rieši pomocou skriptovacích jazykov. Týchto jazykov je mnoho a výber záleží iba na osobnom vkuse.

Jednou z možností je využiť najznámejší [5] hrací skriptovací jazyk a tým je Lua [6]. V hrách ho využívajú na vytvorenie inteligencie pre entity, ale aj pre udalostný systém na vykonanie rôznych akcií. Hlavnou výhodou je rýchlosť vykonávania kódu, široká podpora a využitie v rôznych cudzích komunitách.

Ďalším adeptom môže byť skriptovací jazyk GameMonkey [7]. Oproti predchádzajúcemu je menej poznaný, no rýchlejší a lepší vo viacvláknovom prístupe.

Skriptovanie je taktiež možné prevádzať skriptovacím jazykom JavaScript, kde hlavnou výhodou je množstvo programátorov, ktorí poznajú syntax tohto jazyka. Väčšinou sa ale nevyužíva v hráčskej komunite.

Taktiež je možné vytvoriť svoj vlastný skriptovací jazyk, kde musíme riešiť ako, kedy a čo sa vykonáva pri zadaných skriptoch.

Každý skriptovací je v podstate možný (hlavne kvôli existencii Scripting API pre ľahké pridávanie skriptov), ale vyššie spomínané jazyky sú tými najlepšími kandidátmi.

V našej hre využijeme kombináciu dvoch možností, a to využitie knižnice podporujúcej vykonávanie Lua skriptov, s vytvorením nášho vlastne definovaného jazyka. Pre Lua skriptovací jazyk sa rozhodujeme kvôli jeho rozsiahlej využiteľnosti v hracom priemysle a pre vykonanie rôznych komplikovaných operácií, pričom náš skriptovací jazyk zas vďaka možnosti písania rýchlych skriptov priamo do XML súborov, ktoré vykonajú malé kusy kódu.

2.7 Možnosť modifikácie

Hry poskytujú rôzne možnosti modifikácie obsahu. Väčšinou je nový aj starý obsah uložený v dátových súboroch, prídavných balíčkoch, XML databáze a niektoré aj v textových súboroch.

Dátové súbory sú väčšinou vytvorené, aby nemohol bežný užívateľ ľahko zmeniť obsah v hre. Svoj obsah združujú v súbore alebo v skupine súborov. Na zmenu je nutné poznať štruktúru súboru a vytvoriť externý program, ktorý s ňou dokáže pracovať. Výhodou je existencia celého obsahu v určitej malej množine súborov, čím sa užívateľ nemusí starať o štruktúru hry. Typicky v dátových súboroch nemôžeme zmeniť logiku celej hry, ale iba upravovať to, čo je v hre už definované ako vzhľad, vlastnosti objektov, atď.

Prídavné balíčky dávajú možnosť pridania nového obsahu do hry. Väčšinou sa využíva tento prístup na doplnenie úplne nových funkcií a poskytujú jednoduchý náhľad na rozšírenie hry. Nevýhodou je vytvorenie takej architektúry, ktorá podobné balíčky podporuje.

SQL databáza je dátovým súborom, ktorá ukladá obsah v hre. SQL databáze podporujú SQL príkazy a sú vhodné na ukladanie a rýchle vyhľadávanie dát.

XML je značkovací jazyk udržujúci objektovú hierarchiu. Oproti spomínaným dátovým súborom je možné XML súbory ľahko otvoriť ľubovoľným editorom, pričom povoľujú ľahkú zmenu všetkého, čo je v hre povolené zmeniť. Výhodou je možnosť zmeny a do určitej miery pridať nový obsah, pričom stavba súboru je intuitívna kvôli dodržanej objektovej hierarchii. Jednotlivé názvy tagov sami vysvetľujú, čo menia. Nevýhodou je obstarávanie stavby týchto súborov, ktoré sa pri určitých veľkostiach stávajú neprehľadné na modifikáciu.

Naša hra využije možnosti XML súborov na definovanie základných vlastností hry, ktoré sú ľahko meniteľné. Oproti SQL sme si XML súbory vybrali najmä kvôli jednoduchosti zmien v značkovacích jazykoch, ktoré sa na nich dajú vykonať. Taktiež využijeme možnosť tvorby prídavných balíčkov na pridanie nového obsahu, ktorý sa následne bude dať využiť v týchto XML súboroch.

3 Herný návrh

Táto kapitola sa venuje konkrétnemu popisu herného návrhu nášho projektu, pričom by mal popísať to, čo budeme v ďalšej implementačnej kapitole realizovať na vytvorenie hracieho jadra so všetkými nami chcenými schopnosťami. Tento popis štruktúrujeme do podkapitol, ktoré na seba nadväzujú.

Návrh hry sa opiera o analýzu hry prebranej v predchádzajúcej kapitole, podľa ktorej jednotlivé kapitoly zodpovedajú stanoveným rozhodnutiam a spôsobom, ako by sa mali jednotlivé problémy zrealizovať. V návrhu ďalej predstavujeme, ako spolu navrhnuté spôsoby súvisia.

Hra je navrhnutá tak, aby bolo možné ľahko rozšíriť obsah projektu v čo najviac oblastiach. Týmto spôsobom sa bude dať doplniť prázdne prostredie hry o príbeh, predmety a ostatné prvky. Ľahká rozšíriteľnosť projektu zvyšuje obtiažnosť tvorby hry, no zároveň v nej opadne nutnosť priameho programovania jednoduchých prvkov[4] do kódu.

V neposlednom rade treba pripomenúť, že kvôli podobnosti projektu s hrou Minecraft a vymedzením určitého žánru projektu na SandBox s RPG prvkami sa nám otvára možnosť nahliadnuť do iných titulov, ktoré tieto žánre reprezentujú.

3.1 Samotné hranie

V analýze sme popísali, že sa hra bude odohrávať v reálnom čase. Ďalším stanoveným cieľom je zvládnutá užívateľská interakcia s hrou, kde ako hlavný prostriedok využívame hlavne ovládanie skrze klávesnicu, ako je tomu v konzolových hrách. Ovládanie klávesnicou zahŕňa ako pohyb hráča, tak aj interakcie s entitami. Myš je v tomto prípade iba pasívny nástroj pre interakciu s niektorými UI [5] elementmi. Pravdaže je dôležité ponechať možnosť ľahkého prerobenia, či doplnenia interakcií s myšou pri možnom rozšírení projektu.

3.1.1 Hra v reálnom čase

Takýto prístup je zvolený hlavne kvôli tomu, aby sme čo najvierohodnejšie popísali život postavy v nami vytvorenom svete. Zaručuje nám, že hra je

[4] Tým napríklad myslíme definovanie GUI pre našu hru, či dej hry.

[5] Užívateľské rozhranie.

dynamickejšia, než keby sme zvolili ťahový prístup, ktorý by viedol v takomto spojení žánrov k zníženej hrateľnosti. Nevýhodou je vyššia náročnosť na počítačovú silu nášho počítača kvôli neustálym aktualizáciám objektov nachádzajúcich sa v hre.

3.1.2 Ovládanie

Každému typu hry zodpovedá určitý štýl ovládania, ktorý žáner vyžaduje. Už v analýze sme naznačili, že pre naše potreby budeme využívať hlavne klávesnicu. Pre správne spracovanie vstupu z klávesnice musíme vedieť vstup načítať, pričom načítanie bude oddelené od zbytku hry. Klávesnicou musíme rýchlo reagovať na vstupy a budeme ňou schopný vykonať viacero operácií naraz. Tento typ ovládania je globálny a ľahko zmeniteľný pre celý projekt.

Ovládanie z myši je v hre zastúpene menej a väčšinou je využívané iba pri ovládaní užívateľských komponent. Tiež sme schopný nastaviť, ktoré a kedy sú komponenty schopné reakcií.

3.2 Možnosti sveta

Hru situujeme v 3D nekonečnom svete zobrazovaním 2D plánu, čo kvôli viacerým dôvodom vytvára značné komplikácie. V prvom rade je to nekonečnosť sveta, v ktorom sa postava môže pohybovať. Ďalším je možnosť prechádzať po viacerých poschodiach. Posledným je záruka zaujímavého sveta, ktorý je vždy iný pri priechode tohto nekonečného sveta.

3.2.1 Nekonečné svety

Svet s nekonečnou veľkosťou je jedným spôsobom ako reprezentovať hrací plán. Takýto svet je obmedzený iba kapacitou počítača. Typov hier s takýmto prístupom bolo v histórii pomenej, no v poslednej dobe[6] sa začínajú vytvárať opäť viacej. Pri výbere možnosti, ako si hra poradí s takýmto svetom, treba brať do úvahy, že sme obmedzení určitými zdrojmi, ako je operačná pamäť, disková pamäť, a že by sa ani do jednej z nich nezmestil. Preto takýto svet rozdeľujeme na menšie kúsky, ktoré si môžeme jednotlivo ponechať v pamäti. Nevýhody spočívajú v a) organizovaní pamäti, keďže kúsky sveta sa nedajú skladovať v operačnej pamäti

[6] Tým myslíme časový úsek od vytvorenia Minecraft-u.

všetky naraz; b) lazy[7] generovanie jednotlivých kúskov; c) spôsobe generovania svetov, aby boli stabilné, konzistentné a zaujímavé.

3.2.2 Multi poschodia

Vytváranie mapy, ktorá by simulovala 3D interaktívny svet v 2D vystavilo projekt otázke, ako si poradiť s takýmto problémom tak, aby bola zaručená korektnosť? Riešením je udržovať mapu sveta v operačnej pamäti jednotlivo po vrstvách a pri každej zmene poschodia do nej budeme načítavať príslušné poschodie z disku. Nevýhodou prístupu je veľký počet súborov na disku, ktoré by boli neprehľadné a pri zohľadnení hry (hráč bude pravdepodobne prechádzať poschodiami dosť často) by bol takýto spôsob zbytočne náročný. Ďalším spôsobom je združovať mapu sveta so všetkými vrstvami v operačnej pamäti, pričom zobrazujeme vždy len jedno poschodie, na ktorom sa nachádza hráč. Pri tejto voľbe nedochádza k zbytočným načítavaniam a je ju možné považovať za tú najlepšiu.

3.2.3 Procedurálne generovanie

Termín označuje generovanie terénu pomocou definovaných algoritmov, väčšinou vygenerovaných v priebehu hry. Generovanie sveta takýmto spôsobom zaručí, že dostávame vždy iné mapy s rôznymi prírodnými prekážkami[8]. Generátory terénu môžu byť fraktálové [9], polygonálne [10] s obmenou pre generovanie kockového terénu, a ďalšie... [11].

3.3 Možnosti hráča

Hráč v našom svete spĺňa niekoľko požiadaviek tak, aby bola hra považovaná za skutočnú podobizeň Minecraft-u s prvkami RPG. Z pohľadu RPG je hráč schopný interakcie s cudzími entitami. Taktiež môže spravovať svoj batoh s predmetmi, ktoré môže použiť pre rôzne akcie. Z hľadiska SandBox-u hráč vie vytvárať predmety a je schopný interakcie so svetom formou ťaženia. Tieto žánre majú spoločné množstvo funkcií nad hráčom a ďalšie sú pre normálny chod hry nevyhnutné (pohyb po vrstvách, pohyb medzi vrstvami,...)

---

[7] Lenivé generovanie je spôsob generovania oddielov podľa toho ako ich potrebujeme.

[8] Prírodné prekážky ako sú rokliny, jamy, atď., ktoré musí hráč buď obísť alebo prekročiť, vytvorením výplne, či umelého mostu.

3.3.1 Interakcie s cudzími entitami

Jednou zo základných možností, ktorú je hráč schopný vykonať, je interakcia s cudzími entitami. Takúto interakciu môžeme rozdeliť do dvoch okruhov a to a) konverzovanie; b) súboj. Interakcie sú závislé od atribútov entít, ktoré medzi sebou interagujú.

3.3.1.1 Konverzácie

Konverzovanie je jeden z typov interakcií, ktorý je nevyhnutný pre splnenie princípov každej hry na hrdinov. Konverzácie takým spôsobom rozvíjajú príbeh v hre. Text aktuálneho rozhovoru je možné zobrazovať vo vyhradenom okne, s možnosťou naňho odpovedať. Rozhovory sa v hrách riešia rôzne, no my si volíme, taký ktorý je zaujímavý, logicky usporiadaný, viacnásobne využiteľný a veľmi intuitívny, ako tomu bolo v hre Morrowind.

Morrowind [17] je hra zo série Elder Scrolls, ktorá je známa nielen ako vynikajúca hra, ale aj ako hra so širokými možnosťami rozšírenia rôznych aspektov, medzi ktoré patria aj konverzácie. Tento typ vytvárania rozhovorov obsahuje rôzne skupiny konverzácií, ktoré môžeme priraďovať entitám. Tieto skupiny obsahujú v sebe možné rozhovory vystihujúce túto skupinu s priradenými podmienkami, ktoré určujú za akých okolností sa konverzácia zobrazí. Nevýhodou takéhoto prístupu je ten, že v niektorých prípadoch do takto vytvorených konverzácií nie je vynaložené také úsilie, ako by tomu bolo pri vytváraní rozhovorov priamo pre danú entitu.

3.3.1.2 Súboj

Súboj je ďalším typom interakcie, ktorú hráč môže vykonať. Ako je tomu vo väčšine hier, tak aj tu musí existovať spôsob, ako otestovať svoju postavu. V našej hre sú sily útoku prepočítavané podľa atribútov. Taktiež sa udržuje istý spôsob náhody[9] takýchto útokov s tým, že berieme do úvahy zručnosť hráča. Nakoniec zavádzame do súboja aj možnosť obrániť svoju postavu.

[9] Ako je napríklad simulovanie hodu kockou

3.3.2 Prístup k vlastnostiam, úlohám, batohu

Ďalšie vlastnosti hráča sú a) možnosť pristupovať k informáciám o svojej postave; b) umožnený prístup k aktuálnym úlohám; c) prístup k svojmu batohu s danými predmetmi. Každý prístup bude otvárať špeciálne okno s výpisom.

3.3.3 Vytváranie predmetov

Vytváranie predmetov je prvá z možností hráča, ktorá je prebraná zo SandBox-u. Tvorba predmetov otvára možnosť pospájať rôzne predmety a správnou kombináciou vytvoriť nový predmet, ako je tomu aj v hre Minecraft. Vytváranie môže prebiehať buď priamo v rukách u inej entity alebo v kontajneri, ktorý umožňuje tvorenie. Tvorenie predmetov dodržiava určitú databázu receptov, ktorá bude voľne prístupná skrze XML súbory.

3.3.4 Interakcie so svetom

Interakcia so svetom je taktiež výhradné privilégium Sandbox žánru. Do takýchto úkonov patrí ťaženie, kopanie a pokladanie blokov nachádzajúcich sa vo svete. Je tu zaručená určitá hierarchia odolností týchto blokov pre priechod hráča naprieč nekonečnou hrou. Útočenie na bloky je podobné útokom na entity. Pri zničení sa blok pridáva do batohu ako predmet.

3.4 Príbeh

Príbeh je súčasťou každej hry, no v hrách na hrdinov je až neoddeliteľnou časťou, v ktorej figuruje ako hlavná, tak aj vedľajšie dejové línie. Pre tieto potreby máme vstavanú možnosť, ako ľahko spravovať vytvorený príbeh a zároveň dbať nato, aby možnosti tvorenia príbehu pôsobili intuitívne a štruktúrovane pre ľahšiu modifikáciu či hľadanie prípadných chýb. Vytváranie príbehu môže byť poňaté inak, no pri hrách na hrdinov je riešením vytváranie úloh.

3.4.1 Možnosti tvorenia úloh

Tvorenie úloh je schopné vytvoriť viacero dejových línií, ktoré môžu byť rôzne pospájané so vstavanou možnosťou tvorenia rôznych koncov pre úlohu. Preto sa znova obrátime na spôsob, ako rieši úlohy hra Morrowind.

Úlohy majú stavbu stromu, čím sú rozdelené na menšie pod úlohy. Jedna cesta v takomto úlohovom strome môže znamenať jednu z možných ukončení.

Každý vrchol takéhoto stromu môže mať ďalšie dodatočné podmienky pre pokračovanie prechodu úlohou. Takto vytvorené úlohy majú intuitívne riešenú štruktúru, ku ktorej sa dá ľahko pristupovať pre prípadnú modifikáciu. Nevýhodou a zároveň výhodou takéhoto riešenia je práve táto modifikovateľnosť, pretože neskúsený hráč môže prinajlepšom porušiť štruktúru úloh, no v najhoršom to môže viesť až k nefunkčnosti takejto úlohy.

3.5 Rozšírenie hry

Projekt obsahuje viacero možností rozšírenia našej hry, ako sme už popisovali v analýze. Jeden druh rozšírenia je vysoko úrovňový, a tým dostatočne ľahký, aby bol intuitívny aj pre neprogramátora. Pre doplnenie vyšších funkcií projekt obsahuje aj možnosť balíčkov/pluginov.

Vysoko úrovňový princíp realizujeme prostredníctvom XML súborov, ktoré popisujú každý objekt v hre. Možnosť takéhoto rozšírenia je obmedzený iba na pridania a zmeny takýchto objektov. Rozšíriť sa dá všetko od užívateľského prostredia, rozhovorov, konverzácií, ale aj ďalších základných vlastností v hre. Pri niektorých objektoch je možné priradiť skripty na vykonanie.

Nízko úrovňový spôsob rozširovania projektu je vykonaný pomocou balíčkov/pluginov, ktoré umožňujú pridanie nového obsahu, ktorý sa dá neskôr využiť v XML súboroch.

3.5.1 Skriptovanie

Je jedna z možností, ako rozšíriť projekt. Rozšírením sa v tomto prípade myslí pridanie, ale aj zmena ľubovoľných skriptov, ktoré sa dajú neskôr využiť. Pre korektné skriptovanie je navrhnutý spôsob, ako písať skripty a korektne ich vykonávať. V analýze sme popísali, že budeme využívať dva nástroje na skriptovanie.

Pri lua skriptovaní netreba navrhovať spôsob písania skriptov, keďže skriptovací jazyk má svoju vlastnú dodržovanú syntax [8]. Hra vie, ako načítať skripty, kedy a ako ich vykonať. Taktiež je schopná pracovať s užívateľom vytvorenými knižnicami. Všetky tieto vlastnosti sú obsiahnuté v našom projekte.

Pri návrhu nášho skriptovacieho jazyka je potrebné brať do úvahy to, ako písať a vykonávať nami chcené skripty. Písanie príkazov je navrhnuté s možnosťou

vykonania komplexnejších príkazov a zároveň pôsobí intuitívne. Hra obsahuje sadu možných príkazov, ktorými sa dajú vykonať rôzne akcie, ale aj možnosť pridania nových príkazov.

4 Implementácia a použité technológie

4.1 Programovací jazyk

Celý projekt je programovaný v jazyku JAVA[10]. Tento jazyk poskytuje množstvo výhod, kvôli ktorým sme si ho vybrali. Hlavnou výhodou je prenositeľnosť kódu aj na inšie platformy než len na tej, na ktorej sme ho programovali. Ďalšou je poskytnutie komponent prvkov, s ktorými môžeme pracovať a následne ich zobraziť na hlavnú obrazovku. Spolu s udalostným systémom nám JAVA poskytuje komfort pri práci.

4.2 Spracovávanie hry v reálnom čase

Kvôli spracovaniu v reálnom čase je nutné riešiť, ako a kedy aktualizovať logickú časť hry spolu s vykreslovaciou, tak aby sme zbytočne nepreťažovali počítač. Na rýchle a správne zrealizované spracovanie bez viditeľných[11] prekresľovaní obsahuje stavba projektu určité časti [12] a to a) animačný cyklus, ktorý sa skladá z aktualizácie, renderovania a uspania aplikácie; b) double buffering[12]; c) aktívne renderovanie.

4.2.1 Animačný cyklus

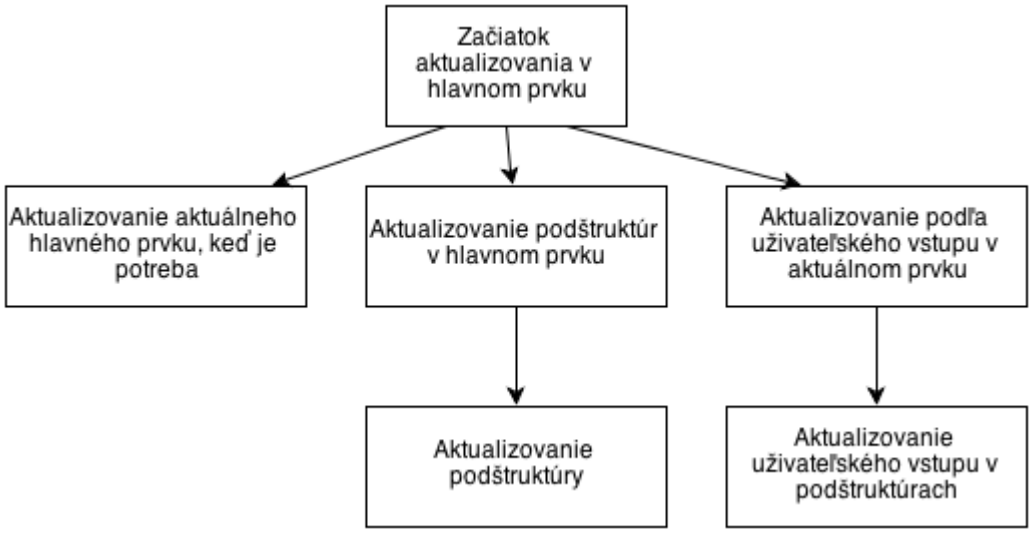
Je hlavný cyklus, ktorý vykonávame dokiaľ hra funguje. Skladá sa z troch úsekov, ktoré vykonávajú svoje príslušné funkcie.

Aktualizácia hry má vykonávať všetky aktualizácie, ktoré v hre prebiehajú. K aktualizáciám, pomimo tých vykonaných na objektoch, patrí aj spracovanie vstupu a komponent nachádzajúcich sa v okne hry. Hlavnou úlohou je prichystať stav na vykreslenie. Aktualizovanie prebieha z najvyššie postaveného prvku v hre, v ktorom začal animačný cyklus a postupne je stromovou štruktúrou aktualizovaný aj zbytok hry (obrázok 4.1). Po správnom vykonaní je projekt prichystaný na vykreslenie.

[10] Pre beh je nutná Java v. 7.

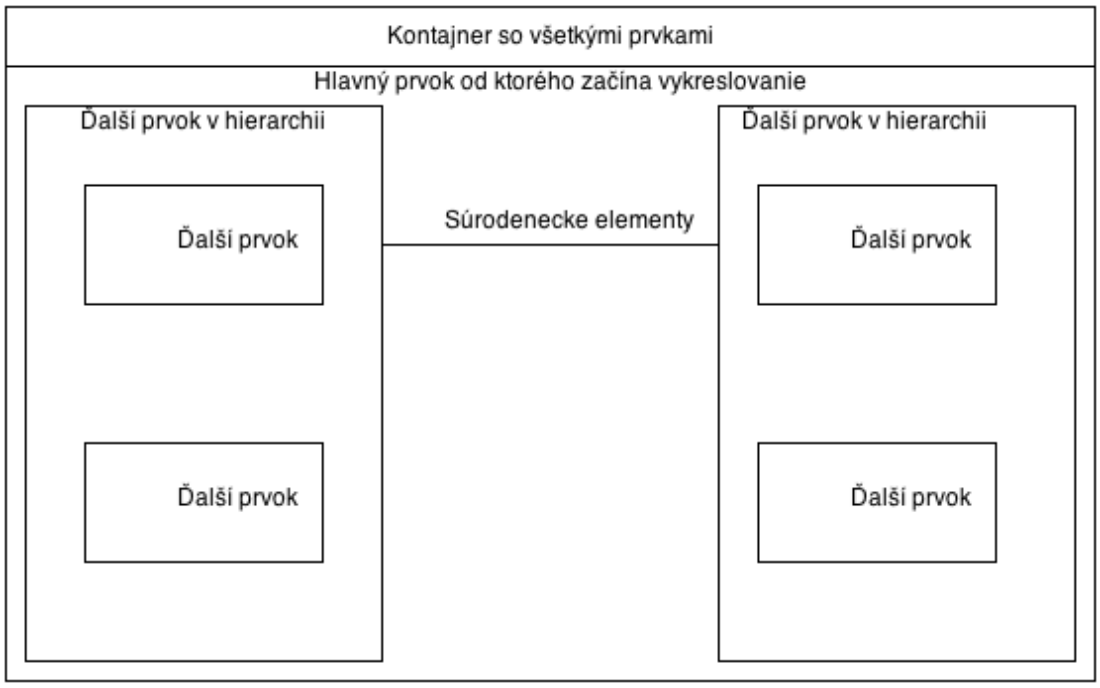
[11] Pri zlom návrhu môže dochádzať pri prekreslovaní k poblikávaniu obrazovky.

[12] Dvojité ukladanie do pamäti.



Obrázok 4.1 : Štruktúra aktualizovania pri prechode animačným cyklom

Renderovanie má za úlohu vykonať vykreslenie prvkov, ktoré sa v hre nachádzajú, v niektorých prípadoch vykreslenie iba tých aktualizovaných. Vykreslenie podobne ako aktualizovanie musí vykonať operácie postupne od hlavného prvku, aby boli zobraziteľné komponenty správne viditeľné (obrázok 4.2). Pri vykresľovaní je dodržanie takéhoto štruktúrovaného vykresľovania ešte dôležitejšie, keďže pri zmýlení poradia môže dôjsť k prekresleniu nejakého prvku, ktorý by mal byť viditeľný.



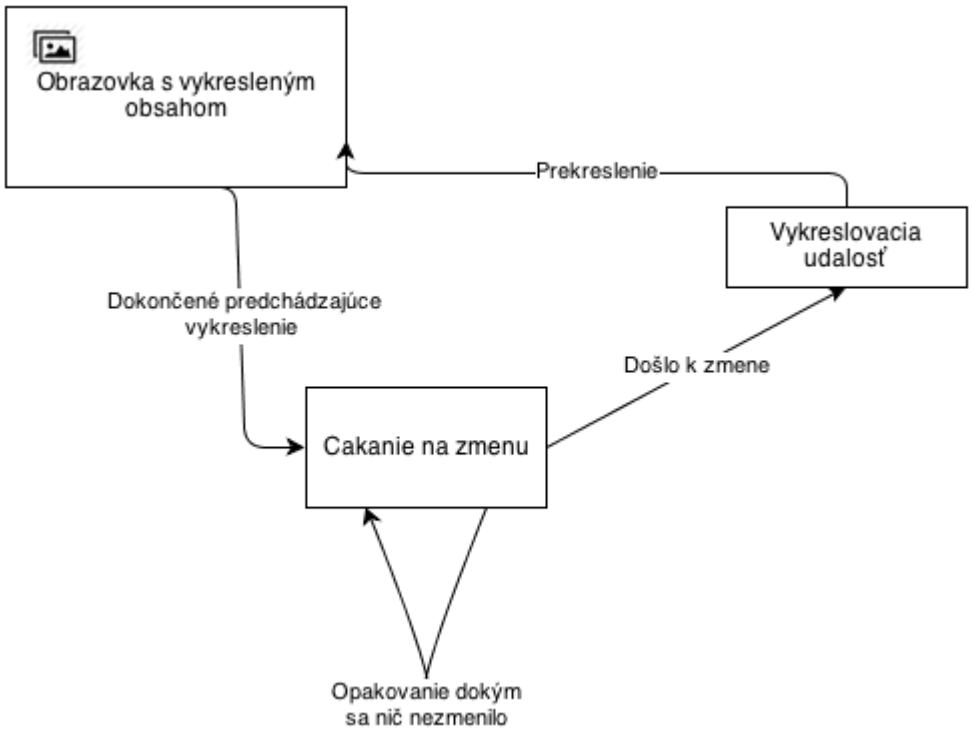
Obrázok 4.2 : Správne zobrazenie komponent pri vykresľovaní

Uspanie aplikácie má dôležitú úlohu v projekte. Pomáha nám obmedzovať počet aktualizácií a vykreslení, ktoré sa vykonávajú, čím uľahčuje počítačový výkon

len na nevyhnutné minimum. Taktiež docielime to, že na každom počítači bude mať hra stabilné FPS[13] a tým pádom sa bude rovnako[14] vykonávať.

4.2.2 Aktívne renderovanie

Vykresľovanie v JAVE používa na prekresľovanie grafických prvkov udalostný systém. Komponenty čakajú až na príchod udalosti, kedy sa vytvorí grafický objekt a zavolajú metódy na vykreslenie. Takýto spôsob renderovania je pasívny a prekreslenie závisí od týchto udalostí. Požiadať o prekreslenie môžeme v samotnom kóde, no môže byť volané aj operačným systémom pri udalostiach ako kliknutie, či zväčšenie (obrázok 4.3). Aktívne renderovanie je pravý opak. Vykreslenie prebieha pravidelne v cykle namiesto toho, aby niekto iný rozhodol, kedy sa bude vykresľovať (obrázok 4.4).



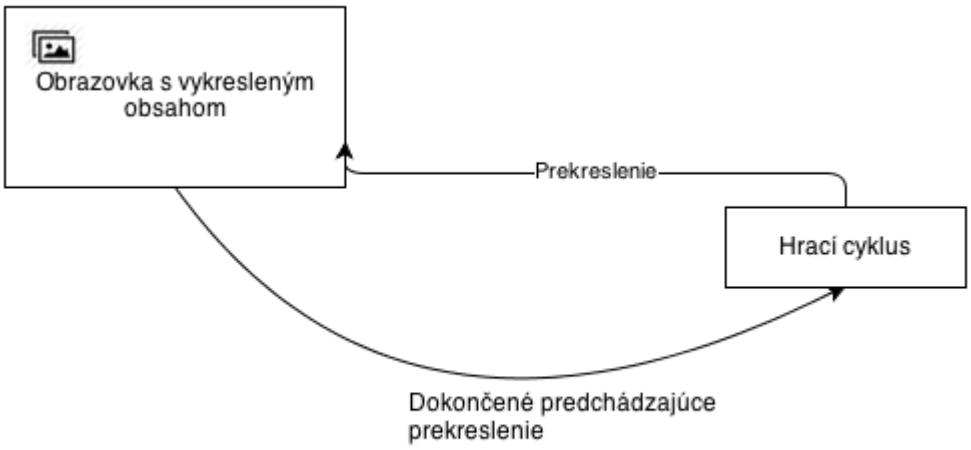
Obrázok 4.3 : Predstava pasívneho vykresľovania

Naše aktívne vykresľovanie je trochu odlišné hlavne kvôli tomu, že v projekte využívame SWING[13] komponenty. Prekresľovanie voláme iba z kódu, no zároveň využívame vstavanú funkciu *repaint* v SWING komponentoch, ktorá zabezpečuje

[13] FPS = frames per second = počet vykreslení vykonaných za sekundu.

[14] Na slabých počítačoch sa zoberie viac z výkonu, na silnejších sa zoberie minimum, no rýchlosť hry bude stale rovnaká.

všetko vykreslenie. Týmto pádom ponechávame udalostný systém, pričom sme prekreslenie zmenili tak, aby dodržovalo aktívne vykresľovanie.



Obrázok 4.4 : Predstava aktívneho vykresľovania

4.3 Spravovanie hracieho plánu

Hrací plán je hlavnou súčasťou hry, vďaka ktorému sa užívateľ naviguje vo svete. Združuje v sebe informácie o aktuálnej pozícií kamery[15], načítaných chunkov (kusov) z nekonečnej mapy, či aktuálnom hracom čase. Plán pracuje s kúskami mapy, ktoré sa v ňom nachádzajú ako aj s objektmi, ktoré sa vyskytujú na mape. Obidve tieto súčasti musí vedieť načítať aj ukladať.

Takýto plán v sebe obsahuje také metódy, ktoré dokážu vykonať aktualizáciu a vykreslenie objektov do plánu. Každá takáto metóda je volaná z príslušného bloku animačného cyklu. Navyše od týchto základných metód obsahuje také, ktoré sprístupňujú hrací plán a jeho objekty aj z iných častí projektu.

4.3.1 Aktualizácia plánu/mapy

Je to základná funkcia plánu, ktorá sa stará o aktualizovanie všetkých objektov, ktoré sa nachádzajú v mape. Taktiež vykonávame aktualizáciu aj na aktuálny čas, častice, ui elementy, či efekty.

Entity sú aktualizované iba vtedy, keď sa nachádzajú v určitej blízkosti hráča, pričom hráča aktualizujeme každým priechodom. Takto zbytočne neaktualizujeme

[15] Kus mapy ktorý vidíme na displeji

niečo, čo na hráča nemá aj tak žiaden vplyv. Kvôli kontrolám entít s ostatnými entitami bude takýto prístup nenáročný[16] na počítač.

Aktualizovanie času je dôležitou súčasťou pri zobrazovaní osvetlenia v mape, a preto ho vykonávame každou sekundou pri priechode. Táto funkciu hry je ďalej súčasťou udalostí pri vytváraní úloh, príbehu, atď.

Častice sú obsiahnuté v každej novodobej hre, a preto vytvárame systém pre spracovávanie častíc. Vytvorené sú väčšinou pri interakciách či akciách hráča s objektmi v hre a aktualizované sú počas celej doby svojej existencie na mape.

Ui elementy sú pomimo SWING komponent možnosťou, ako zobraziť interaktívne okno, s ktorým môže hráč pracovať na vykonanie rôznych funkcií. Aktualizovanie týchto elementov vykonávame iba pri ich existencii, viditeľnosti a aktívnosti. Interakcie v nich podliehajú užívateľskému vstupu.

Efekty sú súčasťou hry, ktoré vykonávajú operácie nad entitami automaticky. Môžu byť pridané do hracieho plánu pri útokoch, či použití nejakého objektu, ktorý v sebe obsahuje efekty. Takto pridaný efekt vykonáva automatické úkony každú sekundu. Pri tejto alternatíve nedochádza k odsynchronizovaniu[17] efektov.

4.3.2 Vykreslenie plánu/mapy

Vykreslenie mapy je nadväzujúcou a neodlučiteľnou súčasťou pri aktualizovaní mapy. Prvou úlohou je vykreslenie pozadia mapy, ktoré bude vykresľovať iba tie časti, ktoré považujeme za viditeľné. Objekty v hracom pláne sú vykreslené v rovnakom poradí ako boli aktualizované, a to podľa ich súradníc vo svete. Nakoniec zostáva vykresliť osvetlenie plánu podľa zaktualizovaného času a vzdialenosti do akej entita dovidí.

4.4 Spravovanie objektov

Ako sme preberali v analýze, hra bude obsahovať viacero druhov objektov ako sú hráč, NPC [18] entity a predmety. Všetky originálne verzie objektov sú

[16] Entity pri aktualizácii kontrolujú, či sa nepretínajú s inými entitami a keďže budeme aktualizovať iba minimálny počet entít a počítame s tým, že entity nie sú vytvorené všetky na jednej kope, tak zložitosť takéhoto úkonu bude menšia než kvadratická.

[17] Efekt by síce vykonal správny počet úkonov, ale časovo by to mohlo trvať dlhšie.

[18] Non Player Creature alebo tiež počítačom ovládané bytosti.

udržované v resource (zdrojových) súboroch. Jednotlivé objekty nachádzajúce sa na pláne, sú kópiami zdrojových objektov s možnou variáciou[19]. Tieto kópie sú následne v aktuálnom hracom pláne aktualizované a vykreslené (obrázok 4.5).

4.4.1 Objekty v zdrojových/resource súboroch

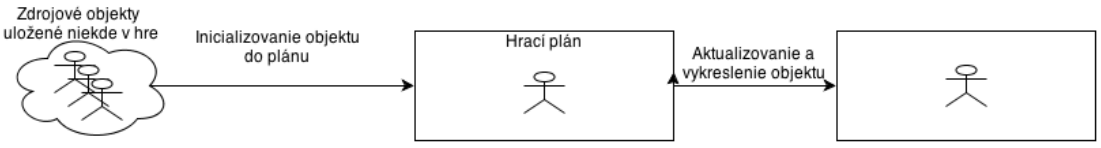
Každý vyššie spomínaný objekt má svoj originálny zdroj, podľa ktorého tento objekt vytvárame. Tieto zdrojové objekty sa nachádzajú v projekte iba raz a inicializujeme ich iba vtedy, keď je to nutné. Tým pádom je prístup k objektom umožnený z každej časti projektu.

Zdrojový objekt popisuje základné vlastnosti objektu ako je typ, obrázok a ďalšie znaky[20], ktoré by mala mať správna RPG hra. Každý musí mať svoje univerzálne popisné číslo a meno pre rozpoznanie pri inicializácií.

4.4.2 Objekty spravované hracím plánom

Hrací plán v sebe združuje objekty, ktoré sme inicializovali zo zdrojového objektu. Nad takýmito objektmi sú vykonávané aktualizácie, za ktorými nasleduje vykreslenie na obrazovku.

Aktualizovanie objektu je vykonávané každým priechodom a po ukončení je zabezpečený korektný stav[21] objektu. Medzi aktualizácie patrí a1) pri hráčovi kontrolovanie užívateľského vstupu a vykonanie operácií podľa kláves; a2) kontrola či sa objekt pohybuje; b) kontrola či sa objekt môže, tam kam chce, pohnúť; c) samotný pohyb podľa určitej inteligencie s nastavením aktuálnych pozícií; d) kontrola okolia objektu s príslušnými operáciami.



Obrázok 4.5 : Predstava inicializovania objektov zo zdrojových objektov do hracieho plánu

[19] Objekt nemusí kopírovať všetky údaje zo zdrojového objektu, ale môže dojsť k menšej obmene ako je napríklad zmena mena.

[20] Znaky ako sila, odolnosť, výdrž, ...

[21] Korektnosť je udávaná podľa toho, čo musí hra obsahovať. Objekty nemôžu prechádzať mimo určenú mapu, nemôže prechádzať nepriestupnými dlaždicami, ...

4.5 Spracovanie užívateľského vstupu

Vstup hráča môže byť dvojakého rázu, tak ako už bolo popísané v analýze hry. Hlavným prostriedkom spracovania je klávesnica, ktorá reaguje na užívateľove stlačenia. Každé stlačenie je zaregistrované a pri aktualizácií hry sa podľa stlačených kláves vykonajú príslušné funkcie. Pri aktualizácií hry berieme do úvahy či je niečo stlačené, čiže aktualizujeme, alebo nie je a neaktualizujeme podľa užívateľského vstupu.

Spracovanie pomocou myši je aktívne iba vtedy, keď chceme, aby bolo aktívne. Taktiež je schopné rozpoznávať počet stlačení a to, na ktoré komponenty reagovať a na ktoré nie.

4.5.1 Globálny ovládač pre vstup z klávesnice

Možnosťou ako spravovať užívateľský vstup je mať jeden globálny ovládač, ktorý naňho reaguje. Takýmto spôsobom neprebieha v projekte viacero spracovávaní vstupu, a tak je riadenie ovládania jednoduchšie. Globálny ovládač je obsiahnutý v každom prvku, ktorý pracuje so vstupom a sprístupňuje mu možnosti, ktoré ponúka.

Ovládač taktiež obsahuje aktuálne stlačené a držané klávesy tak, aby sme vedeli, kedy aktualizovať hru podľa užívateľského vstupu. Tiež obsahuje výpis možných kláves, na ktoré hra reaguje so znakom, ktorý klávesu reprezentuje, pre potrebu zápisu nejakého textu. V neposlednom rade obsahuje definované klávesy obsahujúce klávesu, ktorá ich pomenováva. S definovanými klávesmi hra pracuje priamo a dávajú jej možnosť vytvoriť rôzne ovládacie menu. Hra kontroluje pri testovaní stlačenia kláves iba tieto definované klávesy.

4.5.2 Spracovanie udalostí z myši

Spracovanie udalostí z myši podobne ako pri klávesnici, reaguje na udalosti stlačenia, v tomto prípade na myš. Pri spracovaní rozpoznávame, ktorú komponentu stláčame, počet stlačení, ale aj aktuálnu pozíciu stlačenia. Každé takéto stlačenie na komponentu hry správne vykoná funkciu iba vtedy, keď je komponenta aktívna, viditeľná. Spracovanie udalostí z myši prebieha v samotnom vlákne.

Keďže v našom projekte využívame aj SWING komponenty, tak do nich pridávame poslúchač stlačení, ktorý reaguje na stlačenia a vykonáva príslušný blok

kódu. Samotný SWING a jeho udalostný systém za nás zisťuje, na akú komponentu stláčame a dokonca aj na presnú pozíciu v nej. Komponenty na ktoré stláčame obsahujú príznaky viditeľnosti a aktívnosti. Jediné čo nám zostáva je doplnenie funkcií alebo udalostí, ktoré sa majú pri stlačeniach vykonať. Problém spracovania sa teda zmenšil iba na opatrenie týchto funkcií.

4.6 Spracovávanie a vykonávanie udalostí

Udalosti sú obsiahnuté vo viacerých segmentoch, ako sú reakcie na užívateľský vstup (popísané v predchádzajúcom paragrafe), ale patrí sem aj príbeh, konverzácie, efekty. Vykonávanie udalostí má prístup k väčšine metód obsiahnutých v iných objektoch, aby na nich dokázali vykonať čo najviac zmien. Nato, aby sme korektne vykonávali udalosti, hra vie kedy a akú udalosť treba vykonať. Na plynulé vykonávanie je väčšina udalostí spracovávaná iným vláknom od toho, čo aktualizuje hru. Taktiež je nutné poznamenať, že pre jeden objekt môže byť vykonávaných viacero akcií.

4.6.1 Udalosti v príbehu

Pre príbeh sme pripravili systém, podľa ktorého sa vykonávajú udalosti rôznymi spôsobmi. V príbehu plnia úlohu príkazov na vykonanie špecifických akcií pre daný príbeh. Keďže príbeh je rozdelený do segmentov a medzi segmentmi sa dá prechádzať, udalosti rozdeľujeme podľa toho, kedy sa vykonávajú a to a) na začiatku prepnutia do segmentu, vykonaný raz; b) v dobe segmentu, vykonávaného počas celej doby; c) pri prepnutí na ďalší segment, vykonaný raz.

4.6.2 Udalosti v konverzáciách

Udalosti v konverzáciách plnia dvojitú úlohu a závisí len na umiestnení v XML dokumentoch, akú práve vykonáva. Obidve tieto úlohy sú od seba závislé a bez prvej nie je druhá vôbec braná do úvahy.

V prvom rade je dôležité to, že nie je vhodné ukázať každý rozhovor v konverzačnom paneli. Niektoré sú sprístupnené po určitých akciách a niektoré je vhodné po týchto akciách vymazať. Pre tieto potreby využívame udalosti ako

podmienky, ktoré zisťujú určitý stav[22] a po vykonaní vrátia hodnotu pravda/lož, podľa ktorej rozhodneme, či rozhovor zobraziť.

Druhou úlohou je zaručenie vykonania akcií pri zobrazení konverzácie. Každá konverzácia môže mať svoje akcie a pri rozhovore sa vykonajú len tie, ktoré patria aktuálnej konverzácií.

4.6.3 Udalosti v efektoch

Efekty ako celok dokážu automaticky vykonávať určité akcie na entitách. Pre vykonanie efektu na inom objekte je preto zavedená možnosť spracovania udalosti aj v efektoch. Udalosti sa môžu vykonať iba raz, no nebránime možnosti viacnásobného vykonania.

4.7 Užívateľské prostredie

Užívateľské prostredie hry uľahčuje hráčovi navigáciu v hre. Medzi prvky prostredia by mali patriť menu panely a v nich ďalšie komponenty, ktoré nadobúdajú dvojitého rázu. Kvôli zakomponovaniu SWING-u v našej hre vytvárame také panely s komponentmi, ktoré dokážu so SWING-om pracovať, pričom spracovanie udalostí je v ňom priamo vyriešené. Ďalšou možnosťou je vykresľovanie panelu priamo do grafického kontextu určitej komponenty.

4.7.1 Menu s pomocou SWING

Takéto druhy menu by mali priamo pracovať so SWING lightweight[23] komponentami. Väčšinou je menu popísané ako jedna komponenta (panel), kde obsah v menu ako sú tlačidlá, list, či texty sú ďalšími komponentami (pridanými do panelu). Vytváranie takýchto menu má výhodu v tom, že operácie, ktoré usporiadajú komponentu sú priamo obstarávané samotným SWING-om.

4.7.2 Menu priamo v hre

Tento typ je priamo programovaný v hre a vykresľovaný do grafického kontextu nejakej komponenty. Pridanie takýchto menu je možné jedine pomocou pluginov a sprístupnenie pomocou akcií, ktoré si môžeme dodefinovať. Väčšina

[22] Stav môže naberať rôznych foriem. Môžeme testovať všetko, čo je sprístupnené cez vykonávanie udalostí.

[23] Lighweight komponenty sú celé napísané na JAVA platforme a nezávislé od platformy.

prvkov ako sú texty, obrázky, listy budú musieť byť doprogramované priamo do novovytvoreného menu.

4.8 Možnosti rozšírenia hry

Rozšírenie je možné dvoma spôsobmi. Každý spôsob popisuje, ako ho správne zrealizovať. Rozšírenia sú vytvorené čo najintuitívnejšie a najpochopiteľnejšie s rozumnou štruktúrou, ktorá popisuje základné možnosti daného rozšírenia.

4.8.1 Zdrojové XML súbory

XML súbory sú prvou možnosťou rozšírenia. Táto možnosť je určená pre jednoduché zmeny a pridanie určitého obsahu. XML súborov môže byť viacero a každý z nich naznačuje, čo bude meniť. Pre uľahčenie sú preto súbory situované v adresároch, ktoré určujú, aké dáta budú z XML načítané. XML súborov môže byť v adresári ľubovoľné množstvo, pričom poradie načítania je určené podľa abecedy. Obsah súborov podporuje viacero tagov[24], pričom štruktúra musí dodržiavať DTD[25] schému. Niektoré súbory taktiež podporujú možnosť priradiť objektu určitý skript. Väčšinou je to možné v XML súboroch, ktoré obsahujú tag *action.*

4.8.1.1 Popis formátu XML súborov

XML súborov je mnoho a štruktúra každého z nich popisuje čo najvierohodnejšie, aké možnosti sa pre dané objekty dajú modifikovať. Typy objektov na zmeny sú *ui*, *rozhovory*, *grupy rozhovorov*, *vytváranie podľa receptov*, *entity*, *predmety*, *obrázky*, *hudba*, *úlohy.* Každý zo súborov v sebe vždy obsahuje hlavný prvok a v ňom sadu ďalších prvkov, v ktorých sa už nachádzajú jednotlivé vlastnosti daného objektu (obrázok 4.6). Väčšinou každý definovaný prvok obsahuje svoje vlastné identifikačné číslo. Niektoré z významných vlastností, ktoré podporujú naše XML súbory sú 1) v *ui* XML súboroch podporujeme *template* tag, ktorý zabraňuje zbytočnému výpisu opakujúcich dát; 2) v *ui* súboroch podporujeme možnosť vykonávanie operácii na obrázkoch ako sú rotácie, či zmenenie veľkostí; 3) v *obrázkových* XML súboroch môžeme vykonávať zväčšenia aj zmenšenia; 4)

[24] Tag – Xml/HTML tagy, medzi ktorými sa nachádzajú určité dáta.

[25] DTD – Document Type Definition je jazyk pre popis štruktúry xml dokumentov.

v *entity* aj *predmetových* XML súboroch môžeme vybrať obrázok z listu obrázkov tzv. *sprite sheet*.

```
<?xml version="1.0" encoding="UTF-8"?>
<groups>Hlavný prvok
    <group>Podprvky
        <id>Greetings</id>
        <label>Greetings</label>
        <msgs>id1,id2</msgs>
    </group>
    <group>
        <id>Tutorial2</id>
        <label>What's next?</label>
        <msgs>tut2</msgs>
    </group>
    <group>
        <id>Tutorial1</id>
        <label>I've been sent to you</label>
        <msgs>tut1</msgs>
    </group>
</groups>
```

Informácie o prvku

Obrázok 4.6 : Popis formátu jedného z XML súboru (konverzačných skupín).

4.8.1.2 Skriptovanie

Skriptovanie je možné iba v tých XML súboroch, ktoré majú túto možnosť vo svojom DTD dokumente povolenú, existenciou špeciálneho tagu *action*. Písanie skriptov môže mať dvojitú podobu. Skripty môžeme písať priamo do týchto špeciálnych tagov v XML súboroch. Tieto skripty nazývame *listenery* a sú vykonávané stlačením myši, klávesnice, ale aj pri bežnej aktualizácií úlohy či konverzácie. Tieto skripty majú určitú syntax, ktorú treba dodržiavať. Skript sa skladá z dvoch častí oddelených znakom @. Prvá časť predstavuje pomenovanie knižnice, z ktorej voláme príkaz, ktorého meno je v druhej časti. Na oddelenie dvoch skriptov využívame znak nového riadku. Na zistenie príkazu rozoberáme text s príkazom a určujeme, čo za akciu vykonáme s parametrami, ktoré akcii prislúchajú. Skripty podporujú skoky, uspanie, podmienky, návratové hodnoty a vnorené funkcie.

Ďalším spôsobom je písanie lua skriptov. Jednotlivé skripty združujeme v samostatnom adresári, ktoré môžeme podobne ako *listenery* zavolať z XML súborov. Do tagu vpisujeme cestu k tomuto súboru (obrázok 4.7). Pre správne vykonanie skriptu používame knižnicu *LuaJ*, ktorá priamo zo skriptov umožňuje volať JAVA metódy. Taktiež je možné si dodefinovať svoje vlastné lua pluginy,

ktoré budú podporovať aj operácie so samotnou hrou. Tak bude možné vytvoriť skripty, ktoré by vytvorili napríklad inteligenciu pre entity.

```
<action code="VK_P" type="KEY">DATA@LUA(hello.lua)
game.setValue("ahoj", "test")
empty.log("ahoj")
print( game.value("ahoj") )
game.callListener( "MENUOP@SETMENU(aboutMenu)" )
```

Obrázok 4.7 : XML skript, ktorý volá lua skript

4.8.2 Plugin systém

Druhou možnosťou, ktorou sa dá rozšíriť hra sú súbory nazývané pluginy. Tvorba takýchto súborov je určená pre užívateľov, ktorí sú oboznámení s platformou JAVA. Týmto systémom skôr môžeme pridávať nový obsah do hry než meniť už existujúci.

Plugin môžeme vytvoriť pre rôzne funkcionálne celky, pričom všetky zložky s budú umiestnené v zložke *plugins*. Jednotlivé zložky budú obsahovať dané pluginy pre rôzne funkcie. Plugin systém podporuje a) vytváranie pluginov na generovanie predmetov a mapy; b) vytváranie nových druhov menu; c) vytváranie lua knižníc na rozšírenie príkazov pre lua skriptovanie; d) vytváranie knižníc na rozšírenie príkazov do nášho XML skriptovacieho jazyka; d) vytváranie kresliacich knižníc; f) vytváranie pluginov pre inteligenciu objektov; g) pluginy pre definovanie nových dát použitých v listoch.

Tieto celky väčšinou implementujú alebo dedia od určitej triedy, ktorá tvorí rozhranie pre pluginy tohto typu. Pluginy sú väčšinou načítané do pamäti hry a následne ich využívame až pri ďalších akciách. Každý plugin musí mať v manifeste vypísanú triedu, ktorá tvorí nové funkcie do hry. Pre správnu prácu s nimi je vhodné prečítať si užívateľskú dokumentáciu.

4.8.2.1 Map Plugin

Pomocou týchto pluginov môžeme vytvoriť vlastný generátor máp na obohatenie našej mapy (obrázok 5.2). Hra obsahuje svoj základný generátor v triede *DefaultGenerator,* no pre bohatšie vyzerajúci svet to nestačí. Pluginy sú umiestnené v zložke s ostatnými generátormi. Všetky pluginy tohto typu implementujú triedu *GeneratorPlugin* (obrázok 4.8). Plugin sa doplnením metódy *generate* stáva

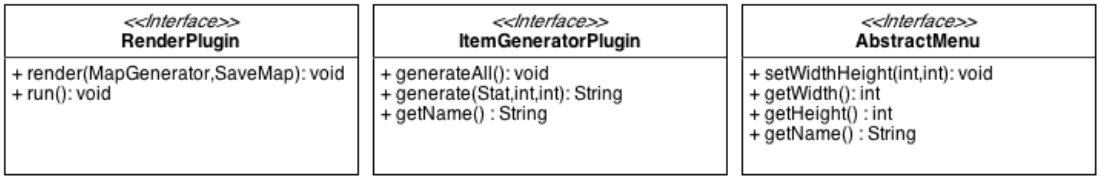
použiteľný na generovanie sveta. Touto metódou sprístupňujeme mapu, do ktorej generujeme nový obsah. Pri generovaní sa postupne zavolá metóda *generate* všetkých načítaných pluginov počínajúc základným generátorom.

4.8.2.2 Item Plugin

Plugin slúži na generovanie náhodných predmetov. Všetky sa nachádzajú v zložke s ďalšími generátormi predmetov. Každý jeden implementuje triedu *ItemGeneratorPlugin* (obrázok 4.8). Na správne generovanie je nutné implementovať metódu *generate*, ktorá predmetu určí špecifické vlastnosti. V triede tiež figuruje metóda *generateAll*, ktorá by mala vytvoriť celý predmet s jeho vlastnosťami. Generovanie predmetov možno v budúcne využiť pri úlohách doplnením skriptovacích príkazov, ktoré takého vytváranie budú podporovať.

4.8.2.3 Menu Pluginy

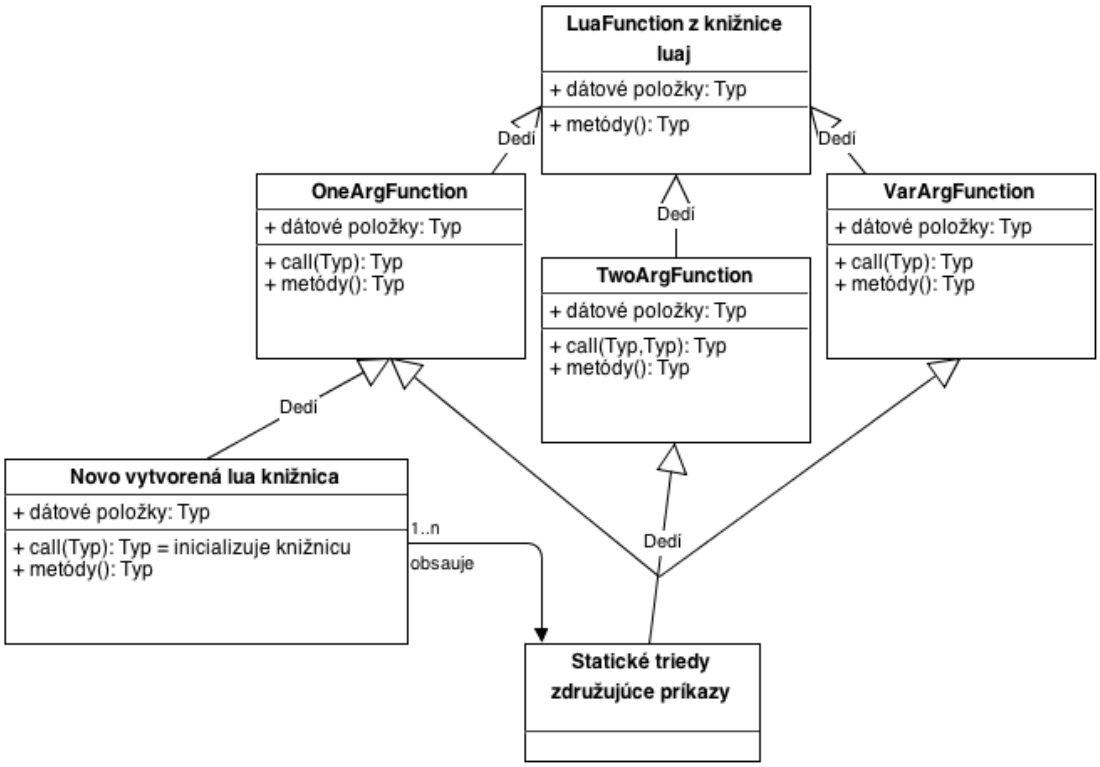
Tento druh pluginov má za úlohu rozšíriť množinu vstavaných menu. Pluginmi môžeme pridať nové SWING, ale aj vstavané menu panely ako je batoh, žurnál,... SWING panely implementujú triedu *AbstractMenu* (obrázok 4.8), ktorá zabezpečuje základné metódy pre správne zobrazenie tohto menu. Vstavané menu implementuje triedu *AbstractInMenu*, kde je nutné implementovať metódy na vykreslenie, aktualizovanie obsahu a interakciu z klávesnici alebo myši. Takto vytvorené panely môžeme zobraziť volaním zo skriptov. Do budúcna je možné vytvoriť taký menu panel, ktorý by podporoval predaj a nákup medzi hráčom a iným objektom.



Obrázok 4.8 : RenderPlugin, ItemGeneratorPlugin, AbstractMenu rozhrania

4.8.2.4 Knižnice Lua

Takéto druhy pluginov rozširujú príkazy pre lua skriptovanie. Na vytvorenie pluginu musíme, popri umiestneniu do adresára s ďalšími lua knižnicami a implementovaniu triedy *ScriptLibraryPlugin*, udržovať určitý spôsob implementácie (obrázok 4.9) pre korektnú funkčnosť [14]. Každý plugin tiež musí dodržovať určité stanovené koncepty z knižnice *LuaJ*.



Obrázok 4.9 : Popis Lua pluginov v závislosti od LuaJ

Pre registrovanie novej knižnice musíme nastaviť nové príkazy definované v statických triedach s prezývkami, ktoré môžeme volať zo skriptov. Registrovanie je vykonané zavolaním metódy *call* na túto knižnicu. Taktiež je nutné priradiť meno, pod ktorým knižnica bude vystupovať (obrázok 4.10). Každý skript je načítaný ako Lua objekt. Vykonanie skriptu je ekvivalentné volaniu metódy *call* na tomto Lua objekte, ktorá vykoná jednotlivé príkazy v skripte. Príkazy sa testujú s názvami zaregistrovaných knižníc. Pri nájdení zhody presúvame volanie do tejto knižnice na definovanú triedu s príkazmi, ktoré v knižnici vystupujú. V tejto triede sa rozhodne, aký príkaz sa vykoná aj s parametrami, ktoré mu patria.

```
@Override
public LuaValue call(LuaValue lv) {
    LuaTable lt = tableOf();

    bind(lt, GameLib1.class, ONEARG_NAMES);
    bind(lt, GameLib2.class, TWOARG_NAMES);

    env.set(LIB_NAME, lt);
    PackageLib.instance.LOADED.set(LIB_NAME, lt);
    return lt;
}
```

Obrázok 4.10 : Správne implementovaná call metóda na registráciu knižnice

4.8.2.5 Knižnice XML skriptov

Knižnice XML skriptov rozširujú existujúcu množinu definovaných príkazov pre XML skriptovanie. Každá takáto knižnica sa nachádza v určitom adresári a implementuje triedu *Listener*. V podkapitole 4.8.1.2 sme popísali, že každý XML skript sa skladá z dvoch častí, kde prvá časť predstavuje pomenovanie knižnice a druhá časť určitý príkaz. Nové pluginy musia kvôli tomu implementovať metódu *getName* na zistenie mena knižnice a *actionPerform* na vykonanie určitého príkazu (obrázok 4.11). Po správnej implementácií je novú knižnicu možné zavolať z XML skriptov.

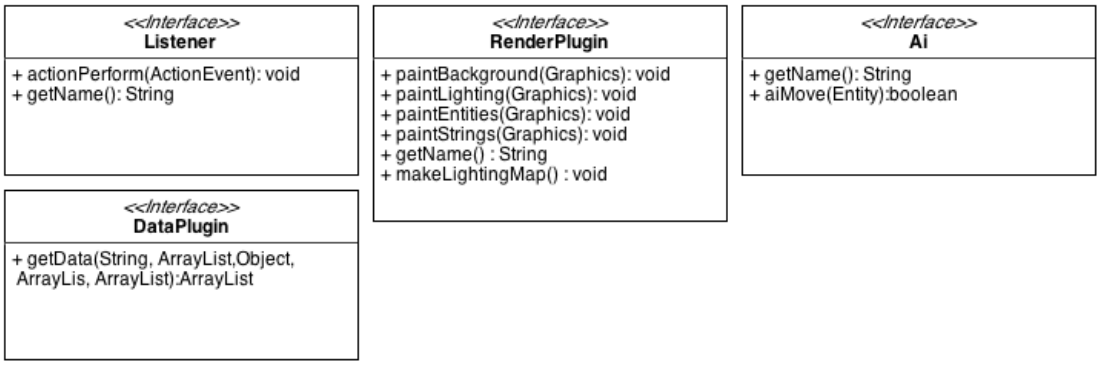
4.8.2.6 Kresliace knižnice

Kresliace knižnice dávajú možnosť prepísať, ako je hra vykresľovaná na obrazovku, a tým vytvoriť krajšiu verziu hry. Knižnice tohto typu implementujú od triedy *RenderPlugin* (obrázok 4.11) a nachádzajú sa v adresári s ďalšími kresliacimi

pluginmi. Rozpoznanie pluginu je závislé na metóde *getName*. Zmenu vykreslenia je možné previesť zavolaním príkazu, ktorý je možný dodefinovať. Knižnica musí ďalej implementovať kresliace metódy *paintBackground, paintEntitiesParticles, paintLighting*, ktoré vykreslia pozadie, objekty, svetlo.

4.8.2.7 Pluginy pre inteligenciu

Pluginmi je možné vytvoriť takú inteligenciu pre objekty, ktorú možno využiť v XML súboroch, kde definujeme tag *ai* s menom novej inteligencie. Pluginy dedia od triedy *Ai* a sú združene v adresári s ostatnými pluginmi. Každá nová inteligencia obsahuje metódu *getName*, ktorou ju rozpoznávame. Metódou *aiMove* pohybujeme a vykonávame rôzne operácie s entitou. Hra je bez týchto pluginov stále hrateľná, lebo v sebe obsahuje základnú inteligenciu *DefaultAi*.



Obrázok 4.11 : Listener, RenderPlugin, Ai, DataPlugin rozhrania

4.8.2.8 Dátové pluginy

Pri vytváraní užívateľského prostredia bolo nutné zabezpečiť, aby sme mohli vytvárať dáta z existujúcich vlastnosti hry, ako je výpis uložených pozícii, výpis akcií nad predmetom, atď. Dátové pluginy rozširujú možnosti týchto definovaných dát. Na vytvorenie pluginu je nutné implementovať triedu *DataPlugin* s metódou *getData*, ktorá vráti požadované dáta. Nové pluginy je ďalej možné použiť v XML súboroch (obrázok 4.12).

```
<listdata data="[#SAVE(NAME,NAME)]">
@Override
public boolean getData(String inf, ArrayList<String> param,
       Object srcObject,ArrayList<ArrayList<Object>> result1,
       ArrayList<Object> result2) {
       switch (Data.valueOf(inf)) {
           case SAVE : {
               infList = Save.getGameSavesParam(param)
```

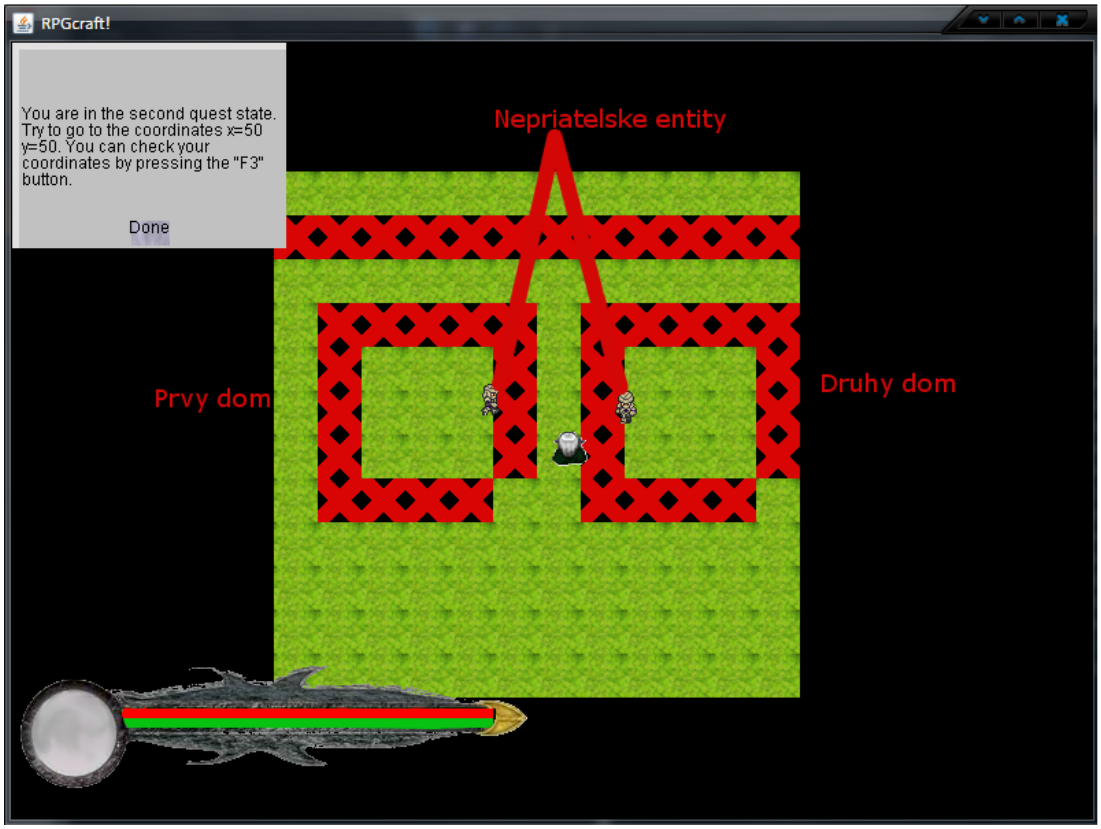
Obrázok 4.12 : Využitie dát v XML súboroch s popisom metódy getData

5 Ukážkové projekty

K projektu sú spolu s prázdnym prostredím priložené aj dve ukážkové hry. Obidve hry obsahujú uvítaciu obrazovku, hlavné menu, menu na vytvorenie a načítanie hry a hracie menu, v ktorom sa odohráva samotná hra. Hry sa odlišujú práve v týchto obrazovkách, ktoré sú vytvorené skrze XML súbory a demonštrujú možnosti definovania užívateľského prostredia. Jednotlivé obrazovky v hrách majú pridané, odobraté či pozmenené komponenty. Na obrazovky je možné klikať, pri niektorých využiť klávesnicu. Hlavná obrazovka s hrou je konceptuálne rovnaká v obidvoch hrách, kde jediné rozdiely predstavujú definované skripty a ako sú definované jednotlivé XML súbory popisujúce konverzácie, úlohy,.... Obidve hry je možné uložiť a vypnúť.

5.1 Ukážková hra

V tejto ukážke je demonštrované, ako sa hra dá navrhnúť. Využívame väčšinu aspektov hry spomínaných v predchádzajúcich kapitolách, ako je definovanie užívateľského prostredia, skripty, pluginy. Medzi panelmi sa dá prechádzať myšou alebo klávesnicou. Panely nemajú postranné testovacie akcie a obsahujú iba tie najzákladnejšie funkcie pre správny chod hry.



Obrázok 5.1 : Obrázok hry bez kreslenia svetla a s ohraničeným terénom

Samotná hracia obrazovka obsahuje hráča s ďalšími vygenerovanými objektmi a dlaždicami, ktoré sú načítané zo zdrojových súborov. Obrazovka je prekrytá svetelnou mapou, ktorú je možno v hre vypnúť. Generovanie terénu podlieha príkazom v plugine vytvárajúcom mapu, ktorý generuje jeden ohraničený blok s dvoma štvorcami s drevenou textúrou predstavujúcimi dva domy (obrázok 5.1). Plugin na generovanie využíva metódu *setTile* na nastavenie dlaždíc (obrázok 5.2) a definovanú metódu *generateHouse* na vytvorenie domov s príslušnou nepriateľskou entitou. Vygenerovaná plocha obsahuje rôzne druhy dlaždíc.

```
public void generateBackground(MapGenerator mapGenerator, SaveMap map) {
    int size = mapGenerator.getSize();
    int id = DefaultTiles.ROCK_ID;
    for (int j = 0; j < size; j++) {
        mapGenerator.setTile(64, j, 0, id);
        mapGenerator.setTile(65, j, 0, id);
        mapGenerator.setTile(64, j, size - 1, id);
        mapGenerator.setTile(65, j, size - 1, id);
        mapGenerator.setTile(64, 0, j, id);
        mapGenerator.setTile(65, 0, j, id);
        mapGenerator.setTile(64, size - 1, j, id);
        mapGenerator.setTile(65, size - 1, j, id);
    }

    generateHouse(mapGenerator, map, 4, 4, 2, 2);
    generateHouse(mapGenerator, map, 4, 4, 8, 2);
}
```

Obrázok 5.2 : Obrázok s popisom pluginu na vygenerovanie terénu

Na obrazovke sa zobrazuje interaktívne okno s aktuálnym textom úlohy, ktorý treba vykonať (obrázok 5.1 vľavo hore). Informácie a štruktúre úlohy je popísaná v súbore *quests.xml*. Úlohy sú rozdelené do stavov, kde sú v tagu *action* definované akcie (obrázok 5.3), ktoré sú vykonané pri prechode úlohou. Všetky udalosti zobrazenia okien a vytvorenia nových objektov sú skripty, ktoré je možné vykonať. Na zobrazenie interaktívneho okna využívame vytvorený príkaz *SHOW_DONE_DIALOG(TEXT)*.

```
    <quest>
        <id>fetch</id>
        <label>Tutorial Quest</label>
        <quest-text>@maintutorial</quest-text>
        <quest-states>
            <state id="0">
                <state-text>@tutorial1</state-text>
                <state-actions>
                    <action type="START">GAME@SHOW_DONE_DIALOG(LOCAL#tutorial1)
DATA@IF(DATA@EQUAL(GAME@CHECK_STATUS(),1),ENTITY@SET_QUESTSTATE(fetch,INT#10))
</action>
                </state-actions>
            </state>
            <state id="10">
                <state-text>@tutorial2</state-text>
                <state-actions>
                    <action type="START">GAME@SHOW_DONE_DIALOG(LOCAL#tutorial2)</action>
                    <action type="THROUGHT">DATA@IF(ENTITY@ISAT_XYZ(INT#50,INT#50,INT#64),ENTITY@S
DATA@TIME_SLEEP(INT#1000)</action>
                </state-actions>
            </state>
            <state id="20">
```

Obrázok 5.3 : Obrázok so štruktúrou definovanej úlohy

Hráč sa môže cez vygenerovaný blok ľubovoľne pohybovať, interagovať s dlaždicami, ale aj s ostatnými objektmi. Taktiež má prístup k batohu, k výrobnému oknu a k ďalším definovaným oknám. Vyrábanie združuje pár receptov, ktoré sa dajú použiť. Recepty sú zhromaždené v *recipes.xml* (obrázok 5.4). Recepty majú svoje identifikačné číslo. Samotný recept je obsiahnutý v tagu *craft-recipe*. Vytvorený predmet z receptu zase v tagu *result*.

```
<recipes>
    <recipe>
        <id>stick</id>
        <shaped>true</shaped>
        <craft-recipe>3,|3,</craft-recipe>
        <result>stick/2</result>
    </recipe>
    <recipe>
        <id>test</id>
        <shaped>true</shaped>
        <craft-recipe>healing1,healing1|healing1,healing1</craft-recipe>
        <result>healing1,tile:1</result>
    </recipe>
</recipes>
```

Obrázok 5.4 : Štruktúra recipes.xml popisujúca dva recepty

5.2 Ukážková testovacia hra

V tejto testovacej hre je oproti predchádzajúcej ukážke demonštrované, čo ďalej ešte hra dokáže. Väčšina užívateľského prostredia v sebe obsahuje rôzne testovacie skripty (obrázok 5.5) definované v XML súboroch, ktoré ukazujú možnosti skriptovania v akcii.

```
        <actions>
            <action code="VK_W" type="KEY">COMPOP@REINITIALIZE</action>
            <action code="VK_A" type="KEY">COMPOP@ADD_COMP_TO(text_ss,MAINCONTAINER)</ac
            <action code="VK_1" type="KEY">GAME@SHOW_DONE_DIALOG(ahojahojahojahojahoja h
            <action code="VK_S" type="KEY">COMPOP@REMOVE_COMP_ALLEXCEPTMENU
DATA@ASSIGN(ahoj,a)
DATA@ASSIGN(VAR#a,b)</action>
            <action code="VK_Q" type="KEY">GAME@SHOW_JOURNAL()</action>
            <action code="VK_P" type="KEY">DATA@LUA(hello.lua)</action>
            <action code="VK_S" type="KEY">COMPOP@SELECTME</action>
            <action code="VK_U" type="KEY">COMPOP@UNSELECTME</action>
            <action code="VK_O" type="KEY">MENUOP@CREATEMENU(about2Menu)</action>
        </actions>
```

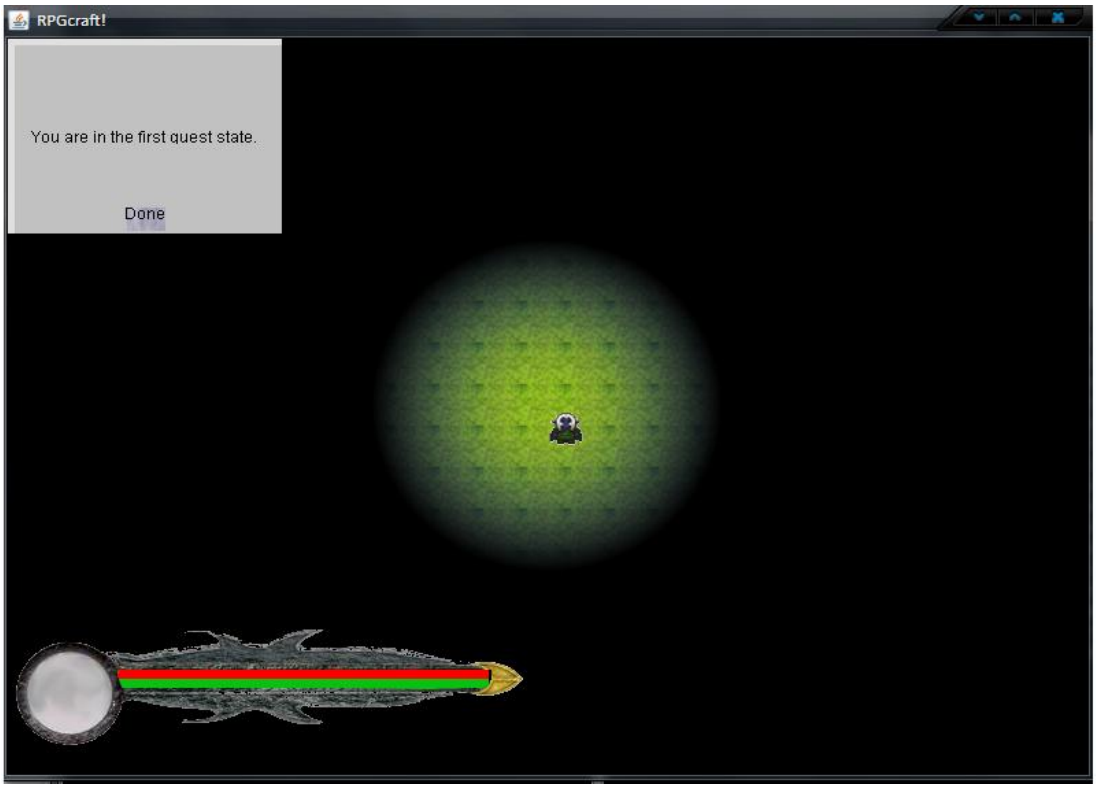
Obrázok 5.5 : Rôzne druhy definovaných XML skriptov

Hra taktiež obsahuje lua knižnicu (obrázok 5.6), ktorú registrujeme pod menom *LIB_NAME="empty"*. V knižnici vytvárame nový testovací príkaz s menom *log*, ktorý dokáže zlogovať určitú správu. Na zavolanie tohto príkazu stačí do lua skriptu napísať *empty.log(TEXT).*

```
public static final String LIB_NAME = "empty";

public static final String[] ONEARG_NAMES = new String[] {
    "log"
};

@Override
public org.luaj.vm2.LuaValue call(org.luaj.vm2.LuaValue lv) {
    LuaTable lt = tableOf();

    bind(lt, GameLib1.class, ONEARG_NAMES);

    env.set(LIB_NAME, lt);
    PackageLib.instance.LOADED.set(LIB_NAME, lt);
    return lt;
}

public static final class GameLib1 extends OneArgFunction {

    @Override
    public LuaValue call(LuaValue lv) {
        switch (opcode) {
            case 0 : {
                LOG.log(Level.INFO, lv.checkjstring());
                return TRUE;
            }
            default : return  NIL;
```

Obrázok 5.6 : Obrázok s popisom Lua pluginu

Hracia obrazovka tiež obsahuje hráča, no kvôli absencii generátora máp nie je plán ohraničený (obrázok 5.7). Pohyb hráča je neobmedzený po nekonečnej mape, ktorá sa neustále opakuje.



Obrázok 5.7 : Obrázok hry bez obmedzení s vykreslením svetla.

6 Príbuzné práce

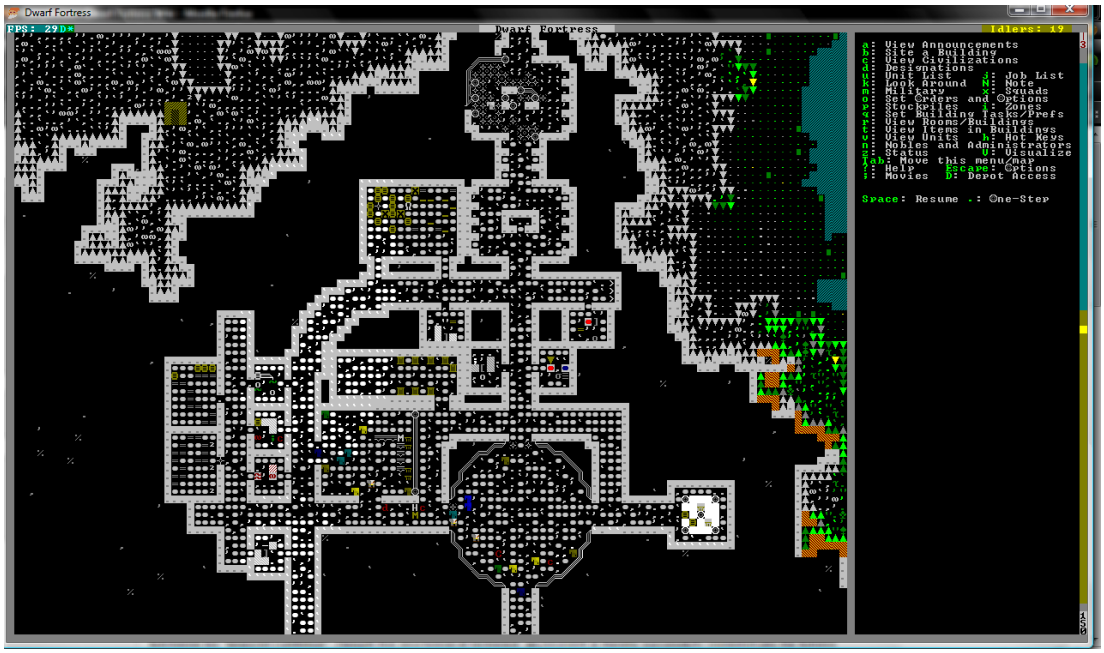
6.1 Minecraft, 2009

Minecraft [2] je prvou známou Sandbox hrou, ktorá má úspech aj v tejto dobe. Základné vlastnosti typické pre Sandbox sú podobné tým, ktoré sme použili v našej hre. V porovnaní s našou hrou je svet zobrazený z prvej osoby v trojrozmernom svete (obrázok 6.1), čo umožňuje ľahšiu navigáciu. Taktiež ako v našej hre poskytuje Minecraft možnosť prestavby a voľného preskúmavania otvoreného sveta. V hre chýbajú hlbšie RPG prvky, ktoré možno nájsť v našej hre, pričom možnosti rozšírenia samotného obsahu jej vcelku chýbajú.

Obrázok 6.1 : 3D Hra Minecraft, ktorá nás ovplyvnila pri tvorbe hry

6.2 Dwarf Fortress, 2006

V porovnaní s našou hrou je Dwarf Fortress [15] Roguelike [26] hra so staviteľskými prvkami situovaná v procedurálne generovanom svete, ktorý je prítomný aj v našej hre. Hra je trojrozmerná, pričom vykresľujeme iba dvojrozmerný plán aktuálneho poschodia (obrázok 6.2) s možným pohybom medzi nimi, čo je presne štýl, akým sa náš projekt uberal. Mapa je vygenerovaná náhodne, oproti našej hre je situovaná iba v určitých obmedzených priestoroch, čím stráca vlastnosť neobmedzenosti sveta.



Obrázok 6.2 : DwarfFortress a hracie okno, v ktorom je vidieť mapa v perspektíve zhora nadol. Grafická stránka hry je jednoduchá.

6.3 ElderScrolls séria (Morrowind)

V porovnaní s našou hrou rieši táto séria všetko, čo by mal správny RPG žáner obsahovať. Oproti nami vytvorenej hry jej chýbajú sandbox prvky a procedurálne generovaný svet, ktorý obmedzuje hru iba na doposiaľ vytvorený svet. V hre sa berie do úvahy aj kúzlenie, ktoré v našej hre úplne chýba. Podobne má hráč prístup k úlohám, batohu a k ďalším prvkom hry, ktoré sú pre žáner typické. S predmetmi môžeme v obidvoch hrách ľubovoľne narábať tak, ako uznáme za vhodné. Užívateľ má taktiež možnosť zmeny obsahu, ako je tomu aj v našej hre.

[26] Je to podžáner RPG charakterizovaný náhodnou generáciou hracieho plánu a ťahovým štýlom.

Oproti balíčkovému systému s XML súbormi má hra priloženú sadu nástrojov na rozširovanie. Obsahuje aj svoj vlastný skriptovací jazyk, avšak chýba jej podpora lua skriptov.

Morrowind [16] ako tretí titul z tejto série obsahuje zo spomínaných mechanizmov zaujímavo, intuitívne a zároveň komplexne riešené rozhovory s úlohami. Tento spôsob zahŕňal spojenie stromovej štruktúry konverzácii aj úloh so samotnými skriptami, ktoré vykonali v hre rôzne udalosti. Takéto riešenie mechanizmov je podobné tomu v našej hre.

Obrázok 6.3 : Logo tretieho titulu zo série ElderScrolls.

7 Záver

Hlavným cieľom tejto práce bolo vytvorenie hry podobnej Minecraft-u s prvkami akčného RPG.

V prvom rade sme vytvorili herný návrh, ktorý bol spracovaný podľa vykonanej analýzy. Určili sme vlastnosti, ktoré hra bude obsahovať a snažili sa, aby bola hrateľná a dávala zmysel v rámci určených žánrov. V hre sme implementovali základné možnosti prebrané zo Sandbox s vlastnosťami, ktoré v sebe združuje RPG žáner. Hlavný hrací engine/hracie jadro, ktoré berie do úvahy všetky vlastnosti hry určené v cieľoch, sme tvorili úplne od začiatku bez využitia podporných knižníc, čo sa nám v konečnom dôsledku aj podarilo. Implementácia je zdokumentovaná a dodržuje pravidlá čistého herného návrhu pre budúce rozšírenia. Hra je v tejto chvíli technicky zrealizovaná skoro kompletne, pričom jediné nedostatky vidíme v neexistencii niektorých funkcií, ktoré by zlepšili pocit z hry, ako je neexistencia procedurálneho generátora máp či v zmene, ako je napríklad vykresľovanie izometrickej mapy.

Taktiež sme určili dodatočné ciele, ktoré obnášajú možnosť modifikácie obsahu hry vo väčšine smeroch a možnosti vytvárania skriptov prostredníctvom knižnice Lua. 1) Pre modifikáciu obsahu sme boli donútení vytvoriť systém, ktorý vie načítať, pracovať a využiť XML súbory ako aj nové balíčky. Takýto systém sme vytvorili, čím sa väčšina obsahu hry presunula do XML dokumentov. Vďaka tomu sa nám uľahčil návrh hry len na zmeny v týchto súboroch. 2) Pre vytváranie skriptov sme zas boli donútení vytvoriť nástroje, ktoré dokážu pracovať s lua skriptami. Popri lua skriptoch sme implementovali možnosť písania vlastných skriptov priamo do XML súborov, ktoré sami o sebe podporujú aj lua skripty.

Do výslednej práce sme naimplementovali všetky nami požadované súčasti, ktoré sme určili v cieli práce. Ľahká rozšíriteľnosť projektu nám uľahčila a urýchlila prácu pri vytvorení ukážkovej hry.

7.1 Budúce práce

Hra obsahuje základné vlastnosti, ktoré by mala podľa cieľov práce obsahovať. Existuje však pár oblastí, ktoré by bolo možné zlepšiť.

Hra, ako sme spomínali v cieľoch práce, neobsahuje automatické generovanie sveta. Možnosť tvorby takéhoto generovania je ponechaná ako jedna z možností,

ktorú možno v budúcnosti zrealizovať generovacími pluginmi. Spolu s generovaním sveta je možné zužitkovať aj generátory predmetov, ktoré sú v tomto stave práce nevyužité.

Ďalšou ponechanou možnosťou je vytvorenie pokročilejšieho vizuálneho spracovania mapy od toho, čo sme vytvorili my. Možným rozšírením je izometrické vykreslenie realizovateľné pomocou kresliacich pluginov. Hru je združovaním 3D sveta možno prepísať na trojrozmerné vykreslenie sveta s príslušnými objektmi, s lepšími efektmi a vykreslením svetla.

K tomu všetkému je tiež možné pridať/pozmeniť nový/existujúci obsah, ako je príbeh, úlohy a ďalšie objekty, ktoré sú uložené v XML súboroch.

8 Zoznam použitej literatúry

[1] IGN : Top100 Games of all Time
http://top100.ign.com/2005/001-010.html

[2] Minecraft
https://minecraft.net/

[3] PlanetMinecraft : PixelArt in Action
http://www.planetminecraft.com/resources/projects/pixel-art/

[4] MinecraftWiki : Awards of 2010
http://www.minecraftwiki.net/wiki/2010_Indie_of_the_Year_Awards

[5] CodePlea : GameScriptingLanguage
http://codeplea.com/game-scripting-languages

[6] Lua : About Lua scripting language
http://www.lua.org/about.html

[7] GmScript : GameMonkey
http://www.gmscript.com/

[8] Lua : Language Manual
http://www.lua.org/manual/5.1/manual.html

[9] GameProgrammer : Fractal Generator
http://www.gameprogrammer.com/fractal.html

[10] Standford : Polygonal Generator
http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/

[11] PCG : Algorithm for Generators
http://pcg.wikidot.com/

[12] DAVISON, Andrew: Killer Game Programming in Java,
1st edition. 2005. ISBN 05-960073-0-2.

[13] SWING : Java LightWeight Components
http://docs.oracle.com/javase/tutorial/uiswing/components/

[14] LuaJ : GettingStarted, Libraries of Java Functions
http://luaj.org/luaj/README.html

[15] DwarfFortress : About Game
http://dwarffortresswiki.org/index.php/Dwarf_Fortress:About

[16] Morrowind : AboutGame

http://www.uesp.net/wiki/Morrowind:Morrowind

9 Zoznam obrázkov

Obrázok 2.1 : Predstava 2D zobrazovania 3D mapy 7
Obrázok 4.1 : Štruktúra aktualizovania pri prechode animačným cyklom 21
Obrázok 4.2 : Správne zobrazenie komponent pri vykresľovaní 21
Obrázok 4.3 : Predstava pasívneho vykresľovania 22
Obrázok 4.4 : Predstava aktívneho vykresľovania 23
Obrázok 4.5 : Predstava inicializovania objektov zo zdrojových objektov 25
Obrázok 4.6 : Popis formátu jedného z XML súboru (konverzačných skupín) 30
Obrázok 4.7 : XML skript, ktorý volá lua skript 31
Obrázok 4.8 : RenderPlugin, ItemGeneratorPlugin, AbstractMenu rozhrania 32
Obrázok 4.9 : Popis Lua pluginov v závislosti od LuaJ 33
Obrázok 4.10 : Správne implementovaná call metóda na registráciu knižnice 34
Obrázok 4.11 : Listener, RenderPlugin, Ai, DataPlugin rozhrania 35
Obrázok 4.12 : Využitie dát v XML súboroch s popisom metódy getData 35
Obrázok 5.1 : Obrázok hry bez kreslenia svetla a s ohraničeným terénom 37
Obrázok 5.2 : Obrázok s popisom pluginu na vygenerovanie terénu 38
Obrázok 5.3 : Obrázok so štruktúrou definovanej úlohy 38
Obrázok 5.4 : Štruktúra recipes.xml popisujúca dva recepty 39
Obrázok 5.5 : Rôzne druhy definovaných XML skriptov 39
Obrázok 5.6 : Obrázok s popisom Lua pluginu 40
Obrázok 5.7 : Obrázok hry bez obmedzení s vykreslením svetla. 40
Obrázok 6.1 : 3D Hra Minecraft, ktorá nás ovplyvnila pri tvorbe hry 41
Obrázok 6.2 : DwarfFortress a hracie okno 42

Obsah priloženého CD

Zložka RPGcraft

NetBeans projekt, v ktorom sa nachádza zdrojový kód hlavnej hry, z ktorej čerpajú ukážkové hry.

Zložka LuaPlugin

NetBeans projekt, v ktorom sa nachádza zdrojový kód vytvorenej lua knižnice.

Zložka GeneratorPlugin

NetBeans projekt, v ktorom sa nachádza zdrojový kód pluginu, ktorý generuje mapu.

Zložka TestGame

Ukážková hra, ktorej obsah demonštruje rôzne testovacie možnosti v hre. Obsahuje zložky s dátami, súborom *RPGcraft.jar* a spustiteľnými súbormi *run.sh* a *run.bat* .

Zložka FullGame

Ďalšia ukážková hra, v ktorej je demonštrované, ako by sa hra dala správne vytvoriť. Obsahuje rovnaké súbory ako je tomu v TestGame.

Bakalárska práca.pdf

Súbor s bakalárskou prácou.

Užívateľská dokumentácia.pdf

Priložený súbor so sprievodným návodom, ako sa dá hra spustiť a následne hrať. Taktiež obsahuje návod na zmenu obsahu a ako sa dá v hre skriptovať.

Zložka JavaDoc

Programátorská dokumentácia vo forme JavaDoc-u vytvorená zo zdrojového kódu. Na vygenerovanie dokumentácie je nutný minimálne Ant 1.8.0.

Readme.txt

Súbor obsahuje to, čo je popísané v tejto kapitole.